57

HUGHES

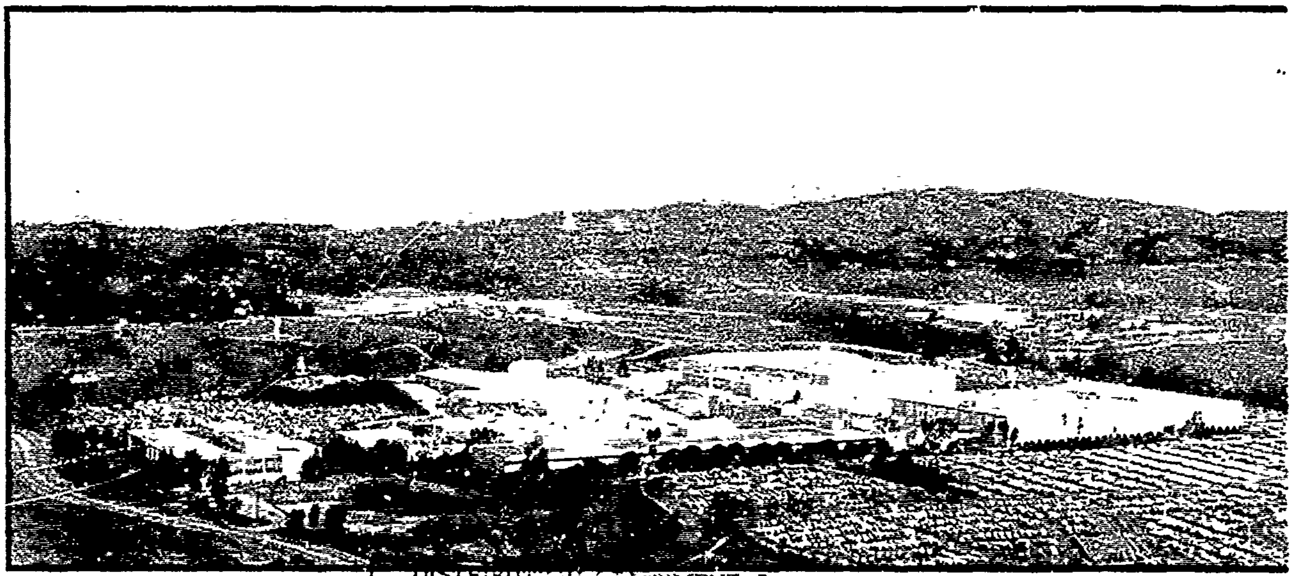
HUGHES AIRCRAFT COMPANY
GROUND SYSTEMS GROUP

Final Report
Appendix

Manufacturing Methods And Technology For Digital Fault Isolation Of Printed Circuit Boards

Project No. R783242

15 NOVEMBER 1980
CONTRACT NO. DAAK 40-78-C-0290



(9) FINAL REP[redacted]RT. APPENDIX

(6) Manufacturing Methods and Technology for
Digital Fault Isolation of Printed Circuit Boards . Appendix .
Project No. R783242

Prepared for
U.S. Army Missile Command
Redstone Arsenal, Alabama 35809

Project Officer: G.D. Little
DRSMI-ETE
(205)-876-3848

(15) Contract No. DAAK 40-78-C-0290
CDRL A004

Prepared by

(12) 100

Hughes Aircraft Company
Ground Systems Group
Fullerton, California 92634

(11) Report Date: 15 Nov[redacted]80

(14) HAC - FR-80-12-975 -APP

172370

RE: Distribution Statement
Basic report is distribution unlimited. Use same statement for appendix to basic per Mr. G. D. Little, Project Officer, DRSMI/ETE

FOREWORD

This report ~~presents~~ the final recommendations and conclusions, with supporting data, ~~resulting~~ from the contract option phase of contract DAAK 40-78-C-0290. ~~It~~ describes the manufacturing technology and test system that will enable detection, identification, and location of digital faults in the advanced missile electronic systems that will be used in the 1980's. Emphasis is placed on the fault diagnosis of large printed circuit boards containing complex hybrid digital microelectronic circuits. The Hughes-enhanced, state-of-the-art, DTS-70 automatic test system installed at Redstone Arsenal as a result of this contract provides the capability to isolate digital faults in such circuit boards to the component level with a test comprehensiveness of 90% or better.

The contract option phase of this project involved the purchase and installation of the DTS-70 system, the selection of the PN-1635972 and the PN-1646178 D/PCBs for testing, the development of generalized test software and the development of the specific hardware and software needed to test these worst-case boards. It also included a successful demonstration of the project's results for interested Department of Defense and industry personnel.

The report concludes with recommendations for the improvement of the DTS-70 System to increase its utility, with recommendations for improving the testability of digital printed circuit boards, and with recommendations for future digital fault isolation studies.

APPENDICES

CONTENTS

APPENDIX A

APPENDIX B

APPENDIX C

APPENDIX A – SOFTWARE

SECTION A. 1

SIGNATURE ANALYSIS SOFTWARE

APPENDIX A – Software
Section A.1 – Signature Analysis Software

A.1.1 SIGNAL TRANSFER FILE TESTING

```
:*
:*     RUN THE INITIALIZING PROGRAM...
:*
: RU, INIT
:*
:*     ASSIGN THE 5004A S.A. AN LU IN SYSTEM...
:*
: SL,35,117
:*
:*     RUN SIGNATURE ANALYZER PROGRAM...
:*
: RU, SCAMPR
:*
:*     EXIT
: TR
```

APPENDIX A – Software
Section A.1 – Signature Analysis Software

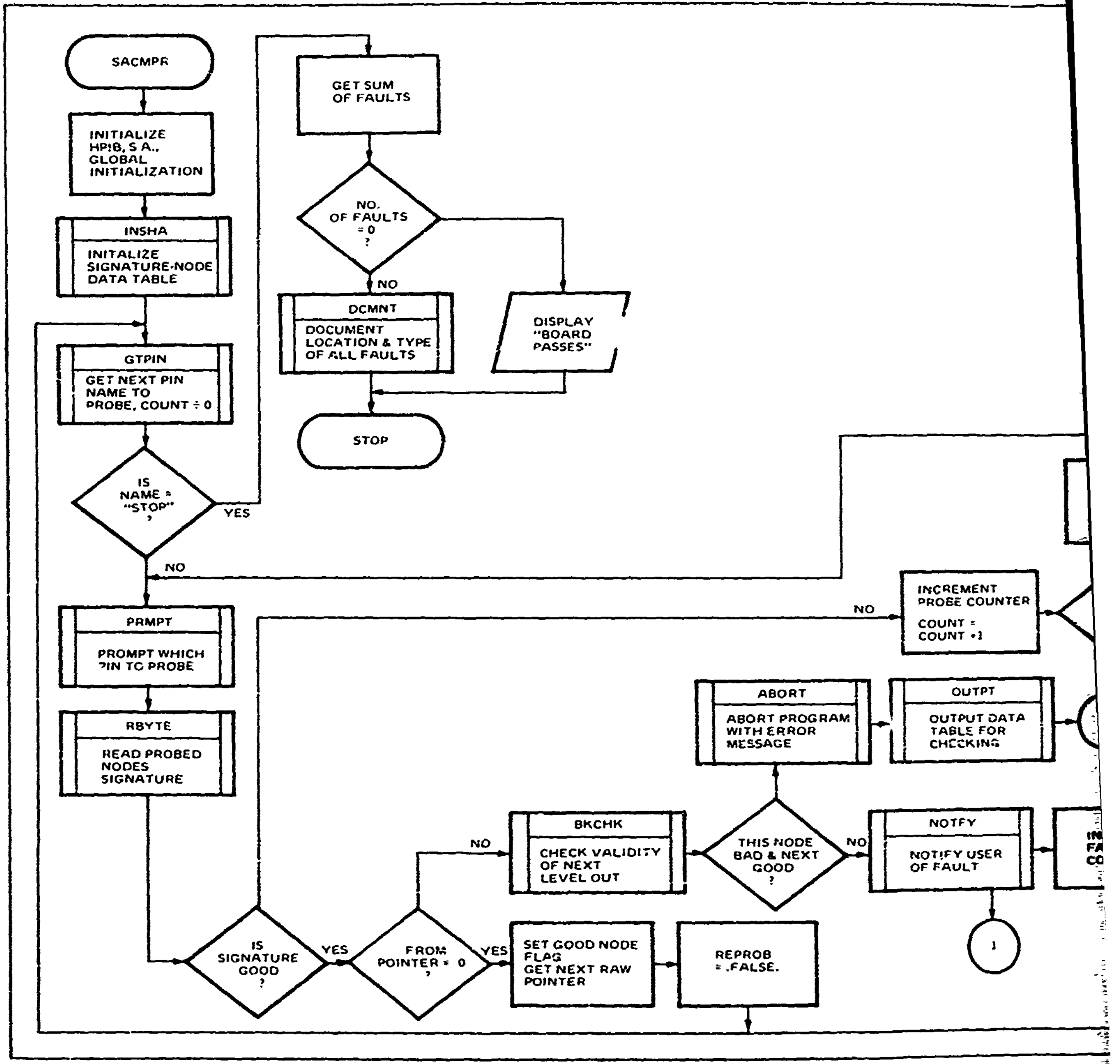
A.1.2 SAFILS SAMPLE LISTING

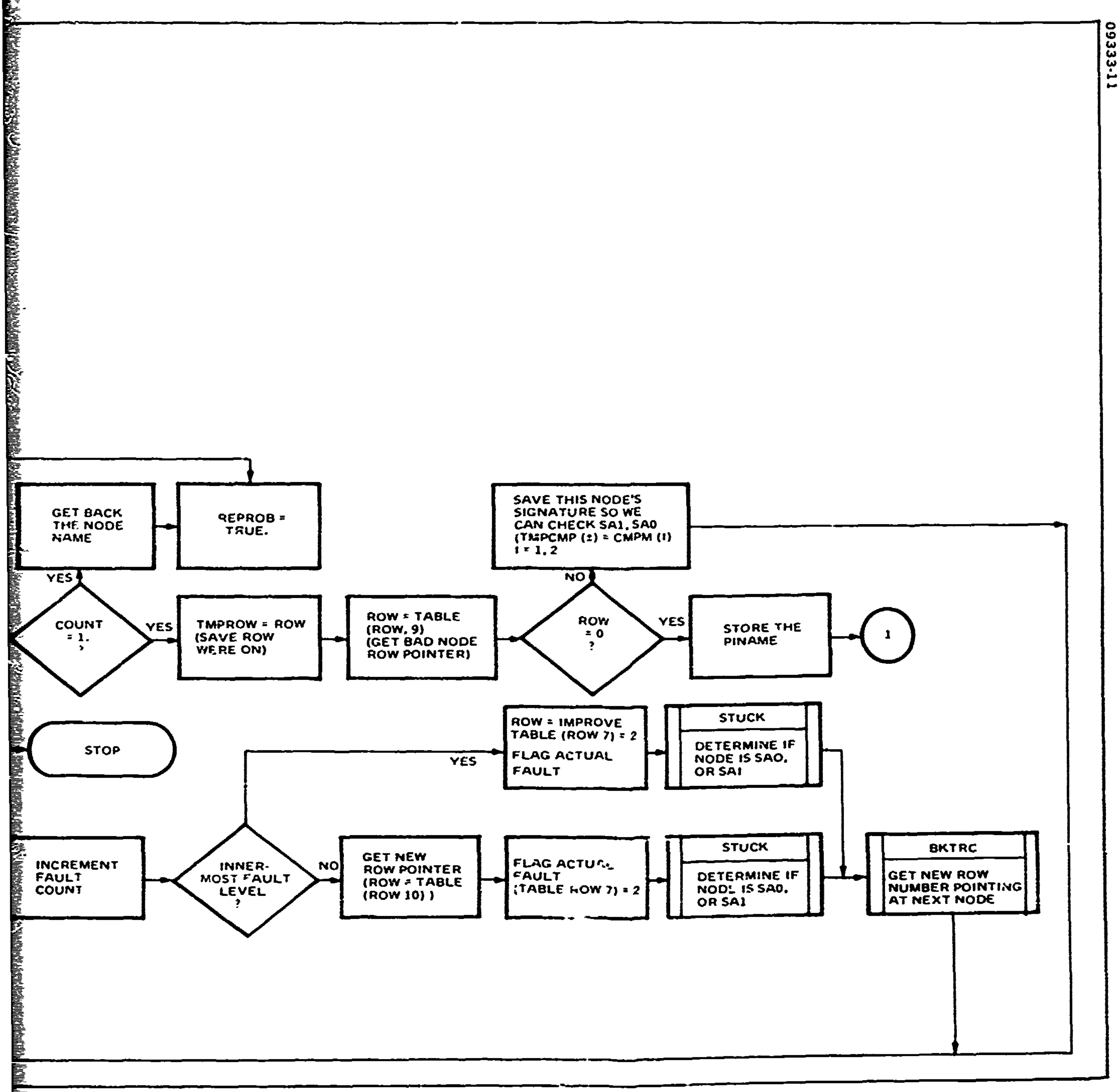
FILE NAME	CR	DESCRIPTION OF CONTENTS
(COL 1-6)	(COL 7-8)	(COL 9-58)
6 chars	2 chars	50 chars
SIG972	19	SIGNATURE ANALYSIS OF ALL ADDRESS&DATA 927 BOARD
CPU972	19	SIGNATURE ANALYSIS OF 8080A CPU CHIP 972 BOARD

END OF FILE

NOTE: The message "END OF FILE" must be placed in the data file starting in the 5th column to show the actual limit of the size of the file.

APPENDIX A – Software
Section A.1 – Signature Analysis Software
A.1.3 – SACMPR Flow Chart


SACMPR
INITIALIZE HPIB, S A., GLOBAL INITIALIZATION
INSHA
INITALIZE SIGNATURE-NODE DATA TABLE
GTPIN
GET NEXT PIN NAME TO PROBE, COUNT ÷ 0
IS NAME = "STOP" ?
YES
NO
PRMPT
PROMPT WHICH ?IN TC PROBE
RBYTE
READ PROBED NODES SIGNATURE
GET SUM OF FAULTS
NO. OF FAULTS = 0 ?
NO
DCMNT
DOCUMENT LOCATION & TYPE OF ALL FAULTS
DISPLAY "BOARD PASSES"
STOP
IS SIGNATURE GOOD ?
YES
FROM POINTER = 0 ?
YES
NO
SET GOOD NODE FLAG GET NEXT RAW POINTER
REPROB = .FALSE.
BKCHK
CHECK VALIDITY OF NEXT LEVEL OUT
THIS NODE BAD & NEXT GOOD ?
NO
ABORT
ABORT PROGRAM WITH ERROR MESSAGE
OUTPT
OUTPUT DATA TABLE FOR CHECKING
NOTFY
NOTIFY USER OF FAULT
1
NO
INCREMENT PROBE COUNTER
COUNT = COUNT +1
IN
FA
CO


GET BACK THE NODE NAME
REPROB = TRUE.
SAVE THIS NODE'S SIGNATURE SO WE CAN CHECK SA1, SA0 (TMPCMP (I) = CMPM (I) I = 1, 2
YES
NO
COUNT = 1. ?
YES
TMPROW = ROW (SAVE ROW WE'RE ON)
ROW = TABLE (ROW, 9) (GET BAD NODE ROW POINTER)
ROW = 0 ?
YES
STORE THE PINAME
1
STOP
ROW = IMPROVE TABLE (ROW 7) = 2
FLAG ACTUAL FAULT
STUCK
DETERMINE IF NODE IS SA0, OR SA1
YES
INCREMENT FAULT COUNT
INNER-MOST FAULT LEVEL ?
NO
GET NEW ROW POINTER (ROW = TABLE (ROW 10))
FLAG ACTUAL FAULT (TABLE ROW 7) = 2
STUCK
DETERMINE IF NODE IS SA0, OR SA1
BKTRC
GET NEW ROW NUMBER POINTING AT NEXT NODE

09333-11

2

A.1.4 – SACMPR LISTINGS

```
SACMPR T=00004 IS ON CR00018 USING 00148 BLKS R=0000

FTN4,L
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C         PROGRAM:   SACMPR
C
C         PURPOSE:   TO DIRECT THE TEST OPERATOR IN THE SIGNATURE ANALYSIS
C                    OF A DIGITAL PCB.
C
C         PROGRAMMER: DAVID S. WAGNER    -   BLDG. 688/T-125
C                     7030 E POTAWATAMI      HUGHES/(213) 802-4190
C                     TUCSON, AZ 85715       FULLERTON, CA 92634
C                     602-296-2760
C
C         DATE:      02 - JUL - 80
C
C         DATA FILE(S):  SAFILS::-18  contains list of available file
C                        names and description of contents
C                        User is queried to enter a data file name from
C                        this list
C         SUBROUTINES USED:
C
C          1. INSHA           - TO INITIALIZE TABLE FOR PROGRAM USE
C          2. GTPIN           - TO GET NEXT PIN TO PBOBE FROM TABLE
C          3. PRMPT           - TO PROMTP TEST OPERATOR TO PROBE PIN
C          4. RBYTE           - TO GET SIGNATURE OF PIN PROBED
C          5. CHECK           - TO COMPARE SIGNATURE WITH CORRECT ONE
C          6. BKCHK           - CHECKS VALIDITY OF PREVIOUS MODE
C          7. NOTFY           - NOTIFY THE OPERATOR OF A FAULT AND LOCATION
C          8. BKTRC           - BACKTRACE TO LAST GOOD PIN LOCATION
C          9. ABORT           - ERROR EXIT FROM THE PROGRAM
C         10. CNVRT           - ROUTINE TO CONVERT PACKED STRING TO UNPACKED
C         11. RCVRT           - ROUTINE TO CONVERT UNPACKED STRING TO PACKED
C         12. NUM             - FUNCTION TO CONVERT HOLLERITH TO INTEGER
C         13. UNPAK           - ROUTINE TO UNPACK AN A2 INTO 2 A1'S
C         14. ZEROR           - ROUTINE TO BLANK UUT ARRAYS
C         15. LENTH           - FUNCTION TO FIND NO. OF CHARACTERS IN STRING
C         16. PACK            - ROUTINE TO PACK 2 A1'S INTO AN A2
C         17. DCMNT           - TO DOCUMENT TEST RESULTS
C         18. STUCK           - TO CHECK IF SA0 OR SA1 FAULT
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C      HIERARCHY OF SUBROUTINES AND FUNCTIONS USED IN SACMPR
C
C
C
C      MAIN
C
C            INSHA
C                  OPEN
C                  READF
C                  CNVRT
C                        NUM
C                        UNPAK
C            GTPIN
```

```
C           PRMPT
C           RBYTE
C           CHECK
C                 RCVRT
C                       LENTH
C                       FACK
C           B'CHK
C           NOTFY
C           STUCK
C           BKTRC
C           ABORT
C           DCMNT
C
C
C
C  FAULT FLAGGING CONVENTION USED IN DATA TABLE
C
C
C  0 -- NO FAULT
C  1 -- A NODE FAULT (NOT NECESSARILY BACKTRACED)
C  2 -- ACTUAL, BACKTRACED FAULT LOCATION
C  3 -- A S:UCK AT 0 FAULT (SA0)
C  4 -- A STUCK AT 1 FAULT (SA1)
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C
C  ARRAYS USED IN THIS PROGRAM:
C
C  ARRAY (200, 10)     Array used as temporary hold for blanking purposes
C  CHARS (2)           Array used in unpacking of input data
C  CMPAR (2)           Used in CoMPARison of read signal and valid signal
C  DUMMY (3)           Array used to hold node name currently under test.
C  FILCHC(35)          Array used to hold data file names and descriotions
C  GND   (2)           Holds the Ground Characteristic Signature
C  HOLD  (4)           Array used to hold probed signature in 4A1 format
C  IBUF  (4)           Temporary storage array
C  IDCB  (144)         Array for Data Control Block for input from file
C  INBUF (400)         Storage array of input data (80 characters packed)
C  INTXT (10)          Packed input text array for reading from %SIGNA
C  IPBUF (10)          Temp. version of above (INTXT)
C  IRBUF (2)           Used for reading from Signature Analyzer
C  IPRMPT(12)          Holds a Logical Unit No. prompt message
C  LINE  (26)          An unpacked array of input text from &SIGNA
C  NAME  (3)           Array to hold the file name (&SIGNA)
C  POSCHR(16)          Array holding 16 possible signature values
C  TABLE (200,10)      The data table holding nodes, signatures, pointers
C  TEXT  (13)          Packed array of input text from &SIGNA
C  TMP   (3)           Temporary array for holding of re-probe node name
C  VCC   (2)           Holds the VCC Characteristic Signature
C
C
C  VARIABLES USED IN THIS PROGRAM:
C
C  COUNT               A flag used to check for necessity of a re-probe
C  FAULTS              A variable used to hold count of faults on board
C  GOOD                Logical variable to flag good or bad signature
C  I                   Index for any and all DO loops
```

```
C  LUCRT                  Logical unit number for the CRT
C  LUSA                   Logical unit number for the Signature Analyzer
C  NUMREC                 Number of records in the Data file of Nodes
C  PRBCNT                 Counts the number of calls to GTPIN (# of probes)
C  REPROB                 Logical variable set TRUE if reprobe is needed
C  ROW                    Pointer into the current row in the TABLE
C  SUM                    The sum total of faults on the board
C  TMPROW                 A temporary holding of the ROW pointer
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C
C   GLOBAL INITIALIZATION
C
C
      PROGRAM SACMP
      IMPLICIT INTEGER (A-Z)
      DIMENSION TABLE (200. 10), ARRAY (200, 10), HOLD (4), POSCHR (16)
      DIMENSION PINAME (3), CMPAR (2), DUMMY (3), INBUF (40), IPBUF (10)
      DIMENSION IBUF (4), IRBUF (2), IDCB (144), CHARS (2), TMPHLD (2)
      DIMENSION INTXT (10),  NAME (3), TEXT (13), LINE (26), TMP (3)
      DIMENSION TAFRAY (15), VCC (2), GND (2), TMPCMP (2), CMPHLD (2)
      DIMENSION FILCHC (35)
      LOGICAL GOOD, VALU, REPROB
      PRBCNT = 0
      ROW    = 1
      LUCRT  = 1
      FAULTS = 0
      NUMREC = 0
      LUSA   = 35
      BELL   = 3400B
      GOOD   = .FALSE.
      VALU   = .FALSE.
      REPROB = .FALSE.
C
C
CC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CC
C
C
CC
CCC CHECK FOR HPIB LU AND INITIALIZE SA ETC...
CC
C
      CALL EXEC (13, LUSA, IEQT5. IEQT4, ISTAT)
      ISCD = 0
      IDVR = 0
      IDWN = 0
C
CC
CCC IF SELECT CODE IS 0  THEN LU IS NOT DEFINED
CC
C
      IF (IAND (IEQT4,77B) .EQ. 0) ISCD = 1
C
CC
CCC DRIVER MUST BE DVR37 FOR HPIB SO CHECK AND SEE...
```

```
CC
C
      IF (IAND (IEQT5/256, 77B) .NE. 37B) IDVR = 1

C
CC
CCC THE LU MUST BE UP SO CHECK IT....
CC
C
      IF (IAND (ISTAT, 100000B) .NE. 0) IDWN = 1
C
CC
CCC CHECK FOR ALL OF THE ABOVE TESTS SUCCESS... (AND HOPE!)
CC
C
      IF (ISCD.EQ.0 .AND. IDVR.EQ.0 .AND. IDWN.EQ.0) GO TO 9
C
CC
CCC IF WE GET HERE, WE UNFORTUNATELY HAVE AN ERROR (OR TWO...)
CC
C
      IF (ISCD .EQ. 1) WRITE (1, 10001)BELL, BELL, BELL
      IF (ISCD .EQ. 0 .AND. IDVR .EQ. 1)WRITE (1,10002)BELL,BELL,BELL
      IF (IDWN .EQ. 1) WRITE (1,10003)BELL, BELL, BELL
      STOP
C
CC
CCC CALCULATE THE HP-IB INTERFACE LU (59310A CARD IN THE COMPUTER)
CC
C
9     LUIB = LUSA - IAND (ISTAT, 77B)
C
CC
CCC SET THE REMOTE ENABLE LINE VIA AN EXEC CALL (VERY IMPORTANT)
CC
C
      CALL EXEC (3, 1600B + LUSA)
C
CC
CCC AND NOW EVERYTHING SHOULD BE READY TO GO... (LET'S HOPE!!!)
CC
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
CC
CCC
CCCC Initialize the data table of known good signatures by reading them in
CCC  from the &SIGNA file...
CC
C
      CALL INSNA  (TABLE  INTX1, IDCB, NAME, TEXT, LINE, NUMREC, FILCHC)
C
CC
CCC
CCCC Find out the next pin that should be probed and get the name of it
CCC
CC
C
```

```
10     CONTINUE
       CALL GTPIN (TABLE, ROW, COLUMN, DUMMY)
       PRBCNT = PRBCNT + 1
C
CC
CCC
CCCC If it's pin name is STOP then we are all done with the whole table
CCC
CC
C
       IF ((DUMMY (1) .EQ. 2HST).AND.(DUMMY (2) .EQ. 2HOP))GO TO 60
       COUNT = 0
C
CC
CCC
CCCC Prompt the operator to prompt it (the pin name)
CCC
CC
C
15     CONTINUE
       CALL PRMPT (DUMMY, REPROB)
       DO 16 I = 1,3
          TMP (I) = DUMMY (I)
16     CONTINUE
C
CC
CCC
CCCC Now get the actual signature probed from the HP-IB interface and the
CCC  Signature analyzer
CC
C
C      CALL INSIG (HOLD)
C      IF (HOLD (1) .EQ. 2HAB) GO TO 60
       CALL RBYTE (LUSA, HOLD. IPBUF)
C
CC
CCC
CCCC Now check to see if it is a good signature or not (GOOD will hold the
CCC  answer either .TRUE. or  FALSE.
CC
C
       CALL CHECK  (TABLE.ROW,GOOD.HOLD,CMPAR,CHARS,VCC,GND,PRBCNT)
       WRITE (1. 10004) (HOLD (I). I = 1. 4)
C
CC
CCC
CCCC If it is bad go to re-probe section of code (GO TO 40)
CCC
CC
C
       IF (.NOT. (GOOD)) GO TO 40
C
CC
CCC
CCCC If it is good with initial probe at board outer edge, then do
CCC  following lines of code, else we must BacKTRaCe
CC
C
          IF (TABLE (ROW, 10) .NE. 0) GO TO 20
C
```

```
CC
CCC Now flag this node as a good node
CC
C
        TABLE (ROW, 6) = 1
C
CC
CCC get the good goto pointer
CC
C
        ROW = TABLE (ROW, 8)
C
CC
CCC We don't want to reprobe
CC
C
        REPROB = .FALSE.
C
CC
CCC loop back and get the next pin that we have to probe
CC
C
        GO TO 10
20    CONTINUE
C
CC
CCC
CCCC  Now see if node level below current was bad by checking previous
CCC   table entry
CC
C
    CALL BKCHK (TABLE, ROW, VALU, PINAME)
    IF (VALU) GO TO 30
C
CC
CCC
CCCC Notify them of the fault and go back through the TABLE to see where
CCC  the next probe should be
CC
C
25      CONTINUE
        CALL NOTFY (PINAME)
        WRITE (1, 10005)BELL
        FAULTS = FAULTS + 1
C
CC
CCC   Just incremented the counter of faults, now flag the fault in the
CCCC  TABLE in one of two types:  it is the furthest in-level node, or
CCC   it is an ordinary fault at any other node:
CC
C
        IF (ROW .EQ. 0) GO TO 28
C
CC
CCC  we get here for a normal node fault (not innermost)
CC
C
C
CC
CCC find out where last node probed was and get it's row number
```

```
CC
C
            ROW = TABLE (ROW, 10)
C
CC
CCC flag an actual fault here
CC
C
            TABLE (ROW, 7) = 2
            CALL STUCK (ROW, TMPCMP, VCC, GND, TABLE)
            GO TO 29
C
CC
CCC  we get here for a fault at the innermost node level
CC
C
28       ROW = TMPROW
         TABLE (ROW, 7) = 2
         CALL STUCK (ROW, CMPAR, VCC, GND, TABLE)
C
CC
CCC  now BackTRaCe through the TABLE to find next node to probe
CC
C
29       CALL BKTRC (TABLE. ROW)
         GO TO 10
30    CONTINUE
C
CC
CCC   NOTE: We should not ever get here as this section deals with the
CCCC        case where a node is good with a previous node being bad. So
CCC         flag an error and abort.
CC
C
      CALL ABORT (TABLE, ROW)
      CALL OUTPT (TABLE, NUMREC)
      STOP
40    CONTINUE
C
CC
CCC
CCCC  Here after a bad probe, so probe again, and then if is still bad
CCC   then follow bad-goto path in list table, else was a misprobe...
CC
C
      COUNT = COUNT + 1
      IF (COUNT .EQ. 1) GO TO 50
C
CC
CCC notify them we are entering backcheck...
CC
C
         WRITE (1, 10006)BELL, BELL
C
CC
CCC save ROW in case new row is 0
CC
C
         TMPROW = ROW
C
```

```
CC
CCC get the new row pointer from the table
CC
C
         ROW = TABLE (ROW, 9)
C
CC
CCC  If ROW is zero then this node is the fault as we cannot backtrace any
CC   further.
C
         IF (ROW .NE. 0) GO TO 46
C
CC
CCC get here if this node is the fault, so store the node name and go
CC  to notification section of the program
C
            DO 45 J = 1. 3
               PINAME (J) = TABLE (TMPROW, J)
45          CONTINUE
            GO TO 25
C
CC
CCC save this bad node's signature case the next node level probe is
CC  good, we have to be able to compare it with VCC and GND characteristic
C   signatures
46       DO 47 I = 1, 2
            TMPCMP (I) = CMPAR (I)
47       CONTINUE
         GO TO 10
C
CC
CCC get node name back in preparation for a Re-Probe
CC
C
50    DO 55 I = 1, 3
         DUMMY (I) = TMP (I)
55    CONTINUE
      REPROB = .TRUE.
      GO TO 15
60    CONTINUE
C
CC
CCC   We get here if we made it thru the table, so now do a checksum on
CCCC  the bad table to see if board (or chip) fails the go/nogo test
CCC
CC
C
      IF (FAULTS .EQ. 0) GO TO 80
         WRITE (LUCRT, 10007)FAULTS
         CALL DCMNT (TABLE, NUMREC, TARRAY)
         STOP
80    WRITE (LUCRT, 10008)BELL, BELL, BELL
      STOP
10001 FORMAT(" !!! LU # GIVEN IS NOT DEFINED",3A1)
10002 FORMAT(" !!! LU # GIVEN IS NOT HP - IB",3A1)
10003 FORMAT(" LU # GIVEN IS DOWN",3A1)
10004 FORMAT(" SIG. IS:",1X,4A1)
10005 FORMAT(" <<< EXITING BackCheck >>>",A1)
10006 FORMAT(" >>> ENTERING BackCheck <<<",2A1)
10007 FORMAT(" BOARD FAILS... THERE WERE ",I5," FAULTS")
```

```
10008 FORMAT(" BOARD PASSES",3A1)
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE ABORT (TABLE, ROW)
C
CC
CCC  THIS SUBROUTINE IS AN ERROR REPORTING EXIT FROM THE PROGRAM
CC
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION TABLE (200,10)
      LUCRT = 32
C
CC
CCC  Write the error message to the output device
CC
C
      WRITE (LUCRT,10) ROW, (TABLE (I), I=1,3)
10    FORMAT(//," ERROR AT ROW=",I4,/," PIN AND CHIP=",3A2)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE BKTRC (TABLE, ROW)
C
CC
CCC   THIS SUBROUTINE TRACES BACK THRU THE TABLE UNTIL IF FINDS THE LAST
CCCC  NON-ZERO GOOD-GOTO ENTRY, SETS ROW TO THAT VALUE AND GETS THE NEXT
CCC   PIN.
CC
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION TABLE (200, 10)
C
CC
CCC Check to see if we now have a non-zero Good Goto pointer
CC
C
10    IF (TABLE (ROW, 8) .NE. 0) GO TO 20
C
CC
CCC No, we don t so decrement the row pointer and check again...
CC
C
         ROW = ROW - 1
         GO TO 10
C
CC
CCC Yes, we do, so set the new ROW to the value of the Good Goto pointer...
CC
C
20    ROW = TABLE (ROW , 8)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE NOTFY (PINAME)
C
CC
CCC   THIS SUBROUTINE NOTIFIES THE TEST OPERATOR OF THE NODE LOCATION OF
CCCC  THE ACTUAL FAULT DETECTED BY THIS SIGNATURE ANALYSIS PROGRAM.
CCC
```

```
CC
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION PINAME (3)
      LUCRT = 1
      WRITE (LUCRT ,5)(PINAME (I), I = 1,3)
5     FORMAT(" LOCATION OF FAULT IS : ".3A2)
C
CC
CCC Check to see if the fault is a VCC or GND fault (special case for each
CC  board)
C
      IF (PINAME (1) .EQ. 2HU4 .AND. PINAME (2) .EQ. 2H5. .AND.
     1    PINAME (3) .EQ. 2H20) GO TO 20
      IF (PINAME (1) .EQ. 2H U .AND. PINAME (2) .EQ. 2H45 .AND.
     1    PINAME (3) .EQ. 2H.2) GO TO 10
         RETURN
10    WRITE (1, 15)
15    FORMAT (" GROUND ERROR ON CPU CHIP, CHECK ORIENTATION")
      RETURN
20    WRITE (1, 25)
25    FORMAT (" VCC ERROR ON CPU CHIP, CHECK ORIENTATION")
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE BKCHK (TABLE, ROW, VALU, PINAME)
C
CC
CCC   THIS SUBROUTINE CHECKS TO SEE IF THE LAST NODE (ACTUALLY NEXT NODE
CCCC  IN CIRCUIT TEST PROGRESSION) WAS BAD, AND THIS ONE IS GOOD, IF THIS
CCC   IS THE CASE THEN IT IS THE NODE IN ERROR SO LOAD IT & SET VALU TRUE
CC
C
      IMPLICIT INTEGER (A-Z)
      LOGICAL VALU
      DIMENSION TABLE (200, 10), PINAME (3)
C
CC
CCC Get the PROM pointer and put it into POINTR
CC
C
      POINTR = TABLE (ROW, 10)
C
CC
CCC Now check node (N-1) to see if BAD flag was set, if so then we're ok
CC
C
      IF (TABLE (POINTR, 7) .NE. 0) GO TO 5
C
CC
CCC If we get here, it means that node N was ok, but N-1 was not, which is
CC  an impossible occurrence, so flag it and return
C
         VALU = .TRUE.
         RETURN
5     VALU = .FALSE.
C
CC
CCC Store the PIN NAME for notifying purposes
CC
```

```
C
      DO 10 J = 1, 3
         PINAME (J) = TABLE (POINTR, J)
10    CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE CHECK (TABLE,ROW,GOOD,HOLD.CMPAR,CHARS,VCC,GND,PRBCNT)
C
CC
CCC   THIS SUBROUTINE CHECKS THE RESULT OF THE SIGNATURE JUST PROBED AND
CCCC  COMPARES IT WITH THE CORRECT SIGNATURE STOPED IN THE TABLE. IF IT
CCC   IS GOOD IT SETS LOGICAL GOOD TO .TRUE.
CC
C
      IMPLICIT INTEGER (A-Z)
      LOGICAL GOOD
      DIMENSION TABLE (200, 10), HOLD (4), CMPAR (2), CHARS (2)
      DIMENSION VCC (2), GND (2)
      PAKLEN = 2
      NUMCHR = 4
C
CC
CCC If PRBCNT is 1 or 2, then this is a VCC or GND characteristic signature
CC  so check it, and if it is, then store it in either VCC or GND
C
      IF (PRBCNT .NE. 1 .AND. PRBCNT .NE. 2) GO TO 19
         IF (PRBCNT .EQ. 2) GO TO 13
            DO 12 J = 1, 2
               VCC (J) = TABLE (ROW, (J + 3) )
12          CONTINUE
            GO TO 19
13       DO 14 J = 1, 2
            GND (J) = TABLE (ROW, (J + 3) )
14       CONTINUE
C
CC
CCC Subroutine RCVRT converts the signature in HOLD (4A1) into 2A2 in
CC  CMPAR so it can compare it with the table
C
19    CALL RCVRT (HOLD, CMPAR, NUMCHR, PAKLEN, CHARS)
C
CC
CCC   NOW WE HAVE THE PROBED SIGNATURE IN CMPAR IN A 2A2 FORMAT (SAME AS IN
CC    TABLE), SO WE CAN NOW COMPARE THEM.
C
      IF((CMPAR(1).EQ.TABLE(ROW,4)).AND.
     +   (CMPAR(2).EQ.TABLE(ROW,5))) GO TO 20
         GOOD = .FALSE.
         TABLE (ROW, 7) = 1 .
         RETURN
20    GOOD = .TRUE.
      TABLE (ROW, 6) = 1
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE RBYTE (LUSH, HOLD, IRBUF)
C
CC
CCC   THIS SUBROUTINE IS THE INTERFACE BETWEEN THE SIGNATURE ANALYZER
```

```
CCCC  AND THE MAIN FORTRAN PROGRAM.  IT GETS THE SIGNATURE AND STORES IT
CCC   IN THE VARIABLE HOLD IN A 4A1 FORMAT.
CC
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION POSCHR (16), HOLD (4), IRBUF (2)
      DATA POSCHR /1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9,1HA,1HC,
     +              1HF,1HH,1HP,1HU/
C
CC
CCC NOW PAUSE TO GIVE THE OPERATOR A CHANCE TO PROBE. WHEN HE(SHE) DOES
CC  THEY THEN ENTER A CARRIAGE RETURN (<CR>) AND IT READS THE SIGNATURE
C
      WRITE (1,1)
1     FORMAT(" ENTER <CR> AFTER PROBE IS SET")
      READ  (1,2)IDUM
2     FORMAT(I2)
      IF (IDUM .EQ. 01 .OR. IDUM .EQ. 10) STOP
      CALL EXEC (1, 2100B+LUSA, IRBUF, -4)
      HOLD (1) = POSCHR (IAND (IRBUF (1), 7400B) /256 + 1)
      HOLD (2) = POSCHR (IAND (IRBUF (1), 17B) + 1)
      HOLD (3) = POSCHR (IAND (IRBUF (2), 7400B) /256 + 1)
      HOLD (4) = POSCHR (IAND (IRBUF (2), 17B) + 1)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE PRMPT (DUMMY, REPROB)
C
CC
CCC   THIS SUBROUTINE PROMPTS THE TEST OPERATOR TO PROBE THE PIN
CC    SPECIFIED BY ALPHANUMERIC VARIABLE DUMMY
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION DUMMY (3)
      LOGICAL REPROB
      LUCRT = 1
      IF (REPROB) GO TO 15
         WRITE (LUCRT,10) DUMMY (I), I= 1,3)
10       FORMAT(/," PLEASE PROBE PIN : ",3A2)
         RETURN
15    WRITE (LUCRT, 20)(DUMMY (I), I = 1, 3)
20    FORMAT(" POSSIBLE MISPROBE",/," PLEASE REPROBE: ",3A2)
      REPROB = .FALSE.
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE GTPIN (TABLE, ROW, COLUMN, DUMMY)
C
CC
CCC   THIS SUBROUTINE GETS THE NEXT ALPHANUMERIC CHARACTER FOR THE
CCCC  PIN NUMBER TO BE PROBED NEXT, AND PLACES IT IN DUMMY
CCC
CC
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION TABLE (200, 10), DUMMY (3)
      DO 10 I = 1,3
         DUMMY (I) = TABLE (ROW ,I)
10    CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE INSHA(TABLE,INTXT,IDCB,NAME,TEXT,LINE,NUMREC,FILCHC)
C
CC
CCC
CCCC  This subroutine reads in the data from the Signature Analysis
CCC   Data file created for each unique PCB, and creates a table in
CC    memory for the rest of SACMPR to utilize.
C
C
C
C
C
C  ARRAYS AND VARIABLES USED IN THIS SUBPROGRAM:
C
C  FILCHC(35)     Array to hold data file name and description of contents
C  INTXT (13)     The data Input Text Buffer
C  IDCB (144)     Input Data Control Block buffer
C  LINE (26)      Holds unpacked version of an input record
C  NAME (3)       Holds the file name of SA data file (in 3A2 format)
C  TABLE (200,10) Table to hold all the input data
C  TEXT (13)      Holding array used by CNVRT to unpack INTXT
C
C  CR             Cartridge Reference Number of Data file
C  TXTLEN         the length of the TEXT array
C  LINLEN         the length of the LINE array
C  IL             # of words to read in a CALL READF record
C  IOPTN          file type parameter to CALL OPEN (not used, hence 0)
C  SECODE         the security code of the SA data file
C  IDCBS          size of the Data Control Block in words
C  IERR           Error parameter on all FMP calls
C  ROW            Row pointer into TABLE
C  NUMREC         Number of Records in the Data file of nodes
C
C
C  EXTERNAL SUBROUTINES AND FUNCTIONS CALLED:
C
C  CNVRT          converts packed text (A2) into unpacked text (A1)
C  NUM            converts hollerith characters into numbers (integer)
C  UNPAK          unpacks an A2 variable into 2-A1 variables
C
C
CC
CCC  INITIALIZATION
CC
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION TABLE (200,10), INTXT (13), IDCB (144), NAME (3)
      DIMENSION TEXT (13), LINE (26), FILCHC (35)
      TXTLEN = 13
      LINLEN = 26
      IOPTN = 0
      SECODE = 2HRT
      IDCBS = 144
      INTXT (3) = 2H
C
CC
CCC Now open up the data-file-name data file
CC
```

```
C
      NAME (1) = 2HSA
      NAME (2) = 2HFI
      NAME (3) = 2HLS
      IL       = 35
      CR       = 18
C
CC
CCC Now actually open up the file
CC
C
       CALL OPEN (IDCB. IERR, NAME)
101    WRITE (1,1)
1      FORMAT(//," FOLLOWING IS LIST OF AVAILABLE FILES, ENTER YOUR",/,
     1" CHOICE FROM THIS LIST.  TO RE-PRINT THE LIST ENTER ?? IN ",/,
     2" RESPONSE TO REQUEST FOR FILE NAME",/,
     3//," FILE NAME",2X,"CR".2X,"DESCRIPTION OF CONTENTS",/,
     4" ---------  --  ----------------------",//,
     5" WHEN ASKED TO HIT <CR> AFTER A PROBE, SHOULD YOU WISH TO ",/,
     6" EXIT THE PROGRAM, TYPE A 1")
       CALL RWNDF (IDCB)
2      CONTINUE
C
CC
CCC Now read one entry from the data-file name data file
CC
C
       CALL READF (IDCB, IERR, FILCHC, IL. LEN)
       IF (IERR .LT. 0) GO TO 9000
       IF (FILCHC (1) .EQ. 2H  ) GO TO 4
          WRITE (1, 3) FILCHC
3         FORMAT(1X,3A2.5X.A2,2X.31A2)
          GO TO 2
4      WRITE (1, 5)
5      FORMAT(//," ENTER FILENAME:")
       READ (1, 6) (NAME (I), I = 1, 3)
6      FOPMAT(3A2)
       IF (NAME (1) .EQ. 2H??) GO TO 101
       WRITE (1, 7)
7      FORMAT(" ENTER CR:")
       READ (1, 8) CR
8      FORMAT(I2)
       IF (CR .LT. 15 .OP. CR. GT. 20) GO TO 4
          CALL OPEN (IDCB, IERR. NAME)
          IF (IERR .LT. 0) GO TO 4
C
CC
CCC we get here if we successfully opened their data file
CC
C
       ROW = 1
C
CC
CCC  Now read the ROWth record...
CC
C
9         CALL READF(IDCB.IERR,INTXT.IL.LEN)
C
CC
CCC  If there is a reading error. go to the error handler
```

```
CC
C
       IF (IERR .LT. 0) GO TO 9000
C
CC
CCC Check for the word END, indicative of EOF
CC
C
       IF (INTXT (3) .EQ. 2HND) GO TO 40
C
CC
CCC Copy the Pin, Chip, and Signature into the data table
CC
C
       DO 10 I = 1,5
         TABLE (ROW, I) =  INTXT (I)
10     CONTINUE
C
CC
CCC Now unpack the record to facilitate conversion into integers...
CC
C
       CALL CNVRT (INTXT, LINE, TXTLEN, LINLEN)
C
CC
CCC put integer value of 6th & 7th column of data file into TABLE
CC
C
       TABLE (ROW, 6) = 10 * NUM (LINE (11)) + NUM (LINE (12))
       TABLE (ROW. 7) = 10 * NUM (LINE (13)) + NUM (LINE (14))
C
CC
CCC Now convert the three pointers (GGTO, BGTO, FROM) into integers.
CC  and store them in their proper location in the TABLE
C
       TABLE (ROW. 8) = 1000*NUM (LINE (15)) +100*NUM(LINE(16))
       TABLE(ROW,8)=TABLE(ROW,8)+10*NUM(LINE(17))+NUM(LINE(18))
       TABLE(ROW,9)=1000*NUM(LINE(19))+100*NUM(LINE(20))
       TABLE(ROW,9)=TABLE(ROW,9)+10*NUM(LINE(21))+NUM(LINE(22))
       TABLE(ROW,10)=1000*NUM(LINE(23))+100*NUM(LINE(24))
       TABLE(ROW,10)=TABLE(ROW,10)+10*NUM(LINE(25))+NUM(LINE(26))
C
CC
CCC Increment the ROW pointer...
CC
C
       ROW = ROW + 1
C
CC
CCC and go back to read another record
CC
C
       GO TO 9
40   CONTINUE
C
CC
CCC we get here upon an EOF
CC
C
     NUMREC = ROW - 1
```

```
      RETURN
9000  WRITE (6,9010)IERR,NAME,CR
9010  FORMAT(" ERROR #",I4,"ON FILE:",3A2,"::',I3)
      STOP
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE CNVRT (TEXT, LINE, TXTLEN, LINLEN)
C
CC
CCC  This subroutine converts packed line in TEXT into an unpacked string
CC   twice as long, in LINE
C
C
C
C VARIABLES AND ARRAYS USED IN THIS SUBROUTINE
C
C
C  CHARS(2)      holds two unpacked characters used in call unpack
C  TEXT(TXTLEN) holds the packed text that needs to be unpacked
C  LINE(LINLEN) holds the unpacked version of TEXT
C  PTR           a horizontal column pointer into LINE
C
C
C EXTERNAL SUBROUTINES CALLED
C
C UNPAK          unpacks two packed characters into two unpacked CHARS
C
      IMPLICIT INTEGER (A - Z)
      DIMENSION CHARS (2), TEXT (TXTLEN), LINE (LINLEN), TMP (2)
      PTR = 0
C
CC
CCC  loop from the beginning of the TEXT to the end
CC
C
      DO 10 I = 1, TXTLEN
         PTR = PTR + 1
C
CC
CCC  unpacking as we go... using routine UNPAK
CC
C
         CALL UNPAK (TEXT (I), TMP)
C
CC
CCC  put the unpacked version into line
CC
C
         LINE (PTR) = TMP (1)
         PTR = PTR + 1
         LINE (PTR) = TMP (2)
10    CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      FUNCTION NUM (CHAR)
C
CC
CCC  This function converts a hollerith represented character in CHAR
CC   into an integer number in NUM
```

```
C
C
C
C  VARIABLES AND ARRAYS USED IN THIS FUNCTION
C
C
C  CHAR      the actual character upon which conversion is to be done
C
C
C  SUBROUTINES AND FUNCTIONS USED BY THIS FUNCTION
C
C
C  CODE      system subroutine needed to do type conversion
C
C
      IMPLICIT INTEGER (A - Z)
C
CC
CCC  we treat a blank as a leading zero, so handle accordingly
CC
C
      IF (CHAR .EQ. 1H ) GO TO 10
C
CC
CCC  call system CODE routine to do type conversion
CC
C
         CALL CODE
C
CC
CCC  "Read" the character out of CHAR into NUM
CC
C
         READ (CHAR, 5) NUM
C
CC
CCC  by the following format
CC
C
5        FORMAT(I1)
         RETURN
10    NUM = 0
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE UNPAK (PAK, CHARS)
C
CC
CCC
CCCC This subroutine unpacks characters stored in variable PAK in an A2
CCC  format, into 2 chars. at A1 each in the array CHARS(1), CHARS(2)
CC
C
      IMPLICIT INTEGER (A - Z)
      DIMENSION CHARS (2)
      CALL CODE
      READ(PAK, 5) CHARS
5     FORMAT(2A1)
      RETURN
      END
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
           SUBROUTINE ZEROR (ARRAY, LENGTH, WIDTH)
C
CC
CCC This subroutine sets ARRAY (LENGTH, WIDTH) to zero
CC
C
           IMPLICIT INTEGER (A-Z)
           DIMENSION ARRAY (LENGTH, WIDTH)
           DO 20 I = 1,LENGTH
              DO 10 J = 1,WIDTH
                 ARRAY (I, J) = 0
10            CONTINUE
20         CONTINUE
           RETURN
           END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
           SUBROUTINE OUTPT (TABLE, NUMREC)
C
CC
CCC This subroutine outputs the TABLE full of nodes and signatures
CC
C
           IMPLICIT INTEGER (A - Z)
           DIMENSION TABLE (200 ,10)
           DO 10 I = 1, NUMREC
              WRITE (6,5)I,(TABLE (I,J),J = 1,10)
5             FORMAT(1X,I3,"!",3A2,"!",2A2,"!",I1,"!",I1,3("!",I4))
10         CONTINUE
           WPITE (6,15)
15         FORMAT(1H1)
           RETURN
           END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
           SUBROUTINE RCVRT (HOLD, CMPAR, NUMCHR, PAKLEN, CHARS)
C
CC
CCC  This subroutine takes the text in HOLD and using subroutine PACK
CC   packs it into CMPAR
C
C
C
C VARIABLES AND ARRAYS USED IN THIS SUBROUTINE
C
C
C  HOLD(NUMCHR)  unpacked HOLD of characters that need packing
C  CMPAR(PAKLEN)  packed CMPAR of characters that came from HOLD
C  NUMCHR          the number of unpacked characters in HOLD
C  PAKLEN          the number of packed characters in CMPAR
C  FLAG            flag to tell if we need to add an extra trailing blank
C  CHARS(2)        array of two unpacked characters from which to pack
C  J               horizontal pointer into the HOLD array
C
C
C
C EXTERNAL SUBROUTINES CALLED FROM THIS SUBROUTINE
C
C
C  PACK          packs two characters in CHAR(1)(2) into PAK
C  LENTH         function to return the number of characters in STRING
```

```
C
      IMPLICIT INTEGER (A - Z)
      DIMENSION HOLD (NUMCHR), CMPAR (PAKLEN), CHARS (2)
      FLAG = 0
C
CC
CCC  get the number of characters in HOLD
CC
C
      NUMCHR = LENTH (HOLD)
C
CC
CCC  find out if it needs a trailing blank before it gets packed
CC
C
      IF (MOD (NUMCHR, 2) .EQ. 0) GO TO 5
C
CC
CCC  here if it needs an extra blank, so set FLAG accordingly
CC
C
         FLAG = 1
5     CONTINUE
      J = 0
C
CC
CCC  loop for the number of characters in HOLD
CC
C
      DO 20 I = 1, NUMCHR
C
CC
CCC  increment horizontal column pointer into HOLD to get next character
CC
C
         J = J + 1
C
CC
CCC  put the first character to be packed into CHARS
CC
C
         CHARS (1) = HOLD (J)
C
CC
CCC  see if we need an extra blank to be put in CHARS (2)
CC
C
         IF ( .NOT. ((I .EQ. NUMCHR) .AND. (FLAG .EQ. 1))) GO TO 10
C
CC
CCC  we get here if the trailing blank is needed
CC
C
            CHARS (2) = 1H
            GO TO 15
C
CC
CCC  increment column pointer to get the next character
CC
C
```

```
10       J = J + 1
C
CC
CCC  put the second character into CHARS
CC
C
15       CHARS (2) = HOLD (J)
C
CC
CCC  and pack the two of them into PAK
CC
C
         CALL PACK (PAK, CHARS)
C
CC
CCC  and then put the packed characters into the array CMPAR
CC
C
         CMPAR (I) = PAK
20    CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      FUNCTION LENTH (STRING)
C
CC
CCC This Function determines the number of characters in the array
CC  STRING.  Maximum length is 80 and if the STRING is blank 0 is returned
C
      IMPLICIT INTEGER (A - Z)
      DIMENSION STRING (80)
C
CC
CCC Set string pointer at rightmost end of array
CC
C
      I = 80
C
CC
CCC now check to see if we are at left edge of array
CC
C
5     IF (I .LT. 1) GO TO 20
C
CC
CCC if were not, then check to see if this character is non-blank
CC
C
         IF (STRING (I) .EQ. 1H ) GO TO 10
C
CC
CCC if it is non-blank, then I is the length of this array
CC
C
            LENTH = I
            RETURN
C
CC
CCC if it is a blank, however, then shift the pointer to the left and
CC  loop again.
```

```
C
10       I = I - 1
         GO TO 5
20    LENTH = 0
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE PACK (PAK, CHARS)
C
CC
CCCC This subroutine packs characters stored in array CHAR(1), CHAR(2) in
CCC  an A1 format each, into the variable PAK in an A2 format
CC
C
      IMPLICIT INTEGER (A - Z)
      DIMENSION CHARS (2)
      PAK = 2H
      CALL CODE
      WRITE (PAK, 5) CHARS
5     FORMAT(2A1)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE DCMNT (TABLE, NUMREC, TARRAY)
C
CC
CCC
CCCC This subroutine Documents all of the faults found in the UUT
CCC
CC
C
C
C
C  VARIABLES AND ARRAYS USED IN THIS SUBROUTINE
C
C  LOOP         Pointer used to loop through the TABLE
C  NUMREC       The number of records in TABLE
C  TABLE        The data Table holding all nodes and signatures, etc.
C  TARRY        The array used to hold the system time and date
C
C  EXTERNAL SUBROUTINES CALLED
C
C  FTIME        System routine to get time and date
C
C
      IMPLICIT INTEGER (A - Z)
      DIMENSION TABLE (200, 10), TARRAY (15)
      CALL FTIME (TARRAY)
      CALL OUTPT (TABLE, NUMREC)
C
CC
CCC Write out heading for error(s) on strip printer (LU = 32)
CC
C
C
CC
CCC LOOP through the TABLE scanning for fault flagged nodes...
CC
C
      DO 20 LOOP = 1, NUMREC
```

```
         IF (TABLE (LOOP, 7) .LE. 1) GO TO 20
            IF (TABLE (LOOP, 7) .NE. 2) GO TO 10
               WRITE (32, 9) (TABLE (LOOP, J), J = 1, 3)
9              FORMAT(" FAULT: ",3A2,"    ")
               GO TO 20
10          IF (TABLE (LOOP, 7) .EQ. 4) GO TO 12
               WRITE (32, 11) (TABLE (LOOP, J), J = 1, 3)
11             FORMAT( " SA0 -  ",3A2,"    ")
               GO TO 20
12          WRITE (32, 13) (TABLE (LOOP, J), J = 1, 3)
13          FORMAT(" SA1 -  ",3A2,"    ")
20    CONTINUE
      WRITE (32,30)
30    FORMAT(" BAD SIGNATURE(S):  ",/," -------------------")
      WRITE (32,35) (TARRAY (I), I = 1, 15)
35    FORMAT(" TIME:    ",1X,5A2,/,1X,10A2,/," --------------------",/,
     +4X,"TEST REPORT    ")
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE STUCK (ROW, TMPHLD, VCC, GND, TABLE)
C
CC
CCC
CCCC This subroutine checks to see if the fault found was a Stuck at 0, or
CCC  Stuck at 1. and if so flags a SA0 with 3 in column 7 of table, or 4 if
CC   it is a SA1
C
      IMPLICIT INTEGER (A - Z)
      DIMENSION TABLE (200, 10), VCC (2), GND (2), TMPHLD (2)
C     WRITE (1,1)TMPHLD,GND,VCC
1     FORMAT(" TMPHLD ",2A2,2X."GND ",2A2,2X,"VCC ".2A2)
C
CC
CCC Check to see if it is a SA1, SA0
CC
C
      IF (TMPHLD(1).EQ.VCC(1) .AND. TMPHLD(2).EQ.VCC(2)) GO TO 20
         IF(.NOT.(TMPHLD(1).EQ.GND(1).AND.TMPHLD(2).EQ.GND(2)))GO TO 30
C
CC
CCC Get here if it is a GND (SA0) fault
CC
C
            TABLE (ROW, 7) = 3
            RETURN
20       CONTINUE
C
CC
CCC Get here if it is a VCC (SA1) fault
CC
C
         TABLE (ROW, 7) = 4
         RETURN
30    RETURN
      END
      END$
```

A.1.5 – SACMPR 8080 A/B MICROPROCESSOR DATA TABLE FORMAT

Row #	Node Name	Good Signature	Good Flag	Bad Flag	Good Go To	Bad Go To	"From" Go To	Element Name
	1 2 3	4 5	6	7	8	9	10	Word #
	1 2 3 4 5 6	7 8 9 10	11 12	13 14	15 16	17 18	19 20	Byte #
1	U 4 5 . 20				0 2	0 0	0 0	
2	U 4 5 . 2				0 3	0 0	0 0	
3	U 4 5 . 1				0 9	0 4	0 0	
4	U 4 5 . 4				0 0	0 5	0 3	
5	U 4 5 . 6				0 0	0 6	0 4	
6	U 4 5 . 9				0 0	0 7	0 5	
7	U 4 5 . 1 4				0 0	0 8	0 6	
8	U 4 5 . 1 5				0 0	0 9	0 7	
9	U 4 5 . 1 7				1 5	1 0	0 0	
10	U 4 5 . 1 8				0 0	1 2	0 9	
11	U 4 5 . 1 9				0 0	1 1	1 0	
12	U 4 5 . 2 5				0 0	1 3	1 1	
13	U 4 5 . 2 6				0 0	1 4	1 2	
14	U 4 5 . 2 7				0 0	0 0	1 3	
15	U 4 5 . 2 8				2 3	1 6	0 0	
16	U 4 5 . 2 9				0 0	1 7	1 5	
17	U 4 5 . 3 0				0 0	1 8	1 6	
18	U 4 5 . 3 1				0 0	1 9	1 7	
19	U 4 5 . 3 2				0 0	2 0	1 8	
20	U 4 5 . 3 3				0 0	2 1	1 9	
21	U 4 5 . 3 4				0 0	2 2	2 0	
22	U 4 5 . 3 5				0 0	0 0	2 1	

APPENDIX A – Software
Section 1 – Signature Analysis Software

A.1.6 – SAMPLE SIGNATURE FILE, 8080 A/B MICROPROCESSOR USING NOOP

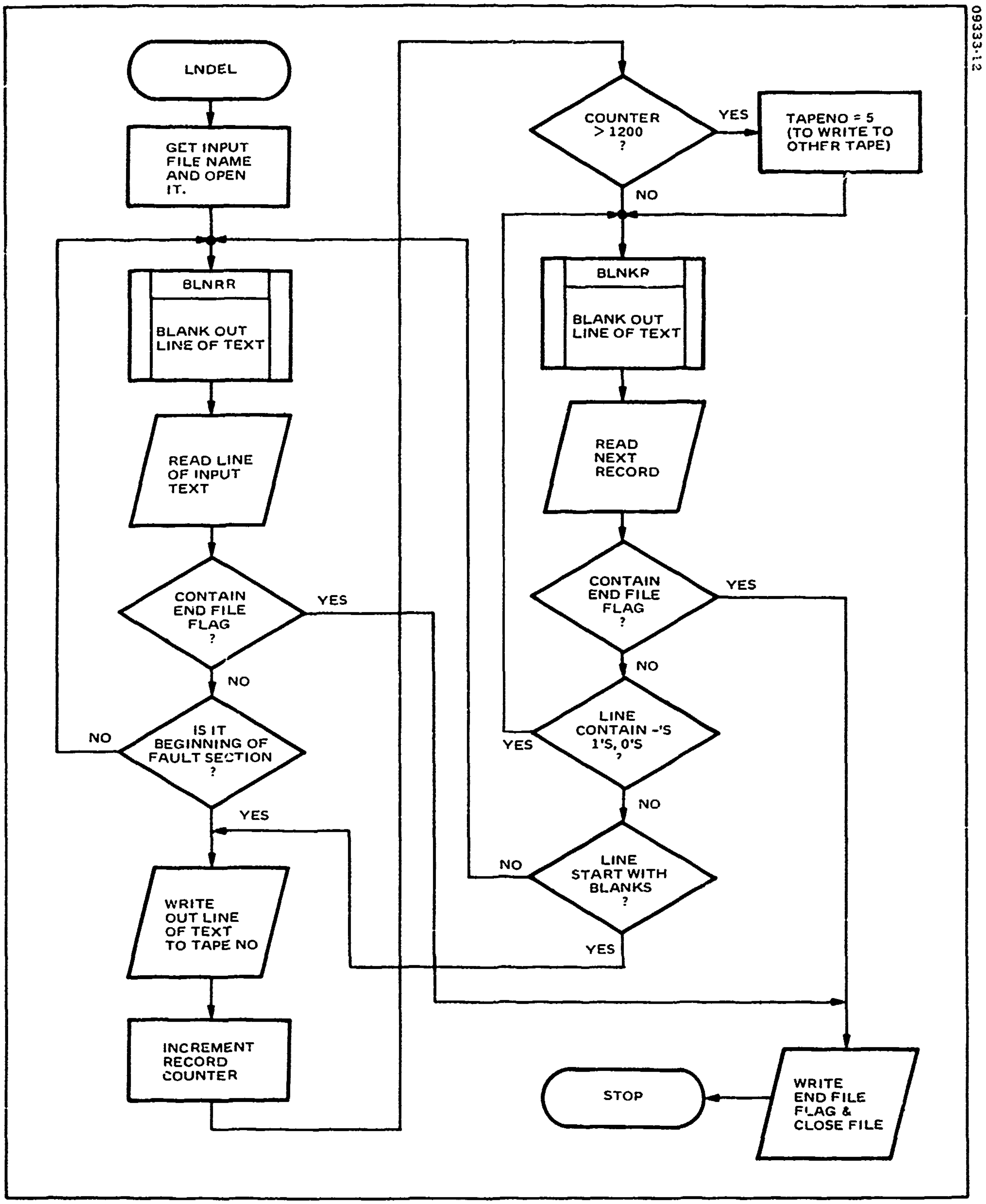
Row #	Node Name						Correct Signature				Good Flag		Bad Flag		Good Go To		Bad Go To		From Go To	
	1		2		3		4		5		6		7		8		9		10	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	U	4	5	.	2	0	7	5	5	U						2		0		0
2		U	4	5	.	2	0	0	0	0						3		0		0
3		U	4	5	.	1	H	H	8	6						4		0		0
4		U	4	5	.	4	7	5	5	U						5		0		0
5		U	4	5	.	6	7	5	5	U						6		0		0
6		U	4	5	.	9	7	5	5	U						7		0		0
7	U	4	5	.	1	4	7	5	5	U						8		0		0
8	U	4	5	.	1	5	7	5	5	U						9		0		0
9	U	4	5	.	1	7	0	0	0	0					1	0		0		0
10	U	4	5	.	1	8	7	5	5	U					1	1		0		0
11	U	4	5	.	1	9	7	5	5	U					1	2		0		0
12	U	4	5	.	2	5	H	3	3	5					1	3		0		0
13	U	4	5	.	2	6	C	1	1	3					1	4		0		0
14	U	4	5	.	2	7	7	0	5	0					1	5		0		0
15	U	4	5	.	2	8	7	5	5	U					1	6		0		0
16	U	4	5	.	2	9	0	7	7	2					1	7		0		0
17	U	4	5	.	3	0	C	4	C	3					1	8		0		0
18	U	4	5	.	3	1	A	A	0	8					1	9		0		0
19	U	4	5.	.	3	2	7	2	1	1					1	0		0		0
20	U	4	5	.	3	3	A	3	C	1					2	1		0		0
21	U	4	5	.	3	4	7	7	0	7					2	2		0		0
22	U	4	5	.	3	5	5	7	7	A					2	3		0		0
23	U	4	5	.	3	6	0	0	0	0					2	4		0		0
24	U	4	5	.	3	7	A	C	9	9					2	5		0		0
25	U	4	5	.	3	8	P	C	F	3					2	6		0		0
26	U	4	5	.	3	9	1	1	8	0					2	7		0		0
27	U	4	5	.	4	0	8	9	F	L					2	8		0		0
28	S	T	O	P	S	T														
29	E	N	D	E	N	D														

NOTE: The data are taken for NOOP Program with the START and CLOCK switches in the IN, and the STOP switch in the OUT position at the HP5004A Signature Analyzer front panel.

SECTION A.2

LNDEL

A.2.1 – LNDEL FLOW CHART



```
      DIMENSION INTXT (18), NAME (3), IDCB (144)
      DATA EOF/2HEF/
      TAPENO = 4
      IL = 18
      COUNT = 0
C
CC
CCC Prompt to get the input file name
CC
C
      WRITE (1,1)
1     FORMAT (" ENTER INPUT FILE NAME:")
      READ (1,2)NAME
2     FORMAT(3A2)
C
CC
CCC Open the input file name, if an open error, flag it and stop
CC
C
      CALL OPEN (IDCB, IERR, NAME)
      IF (IERR .LT. 0) GO TO 9000
C
CC
CCC Blank out the current line of text
CC
C
60    CALL BLNKR (INTXT)
C
CC
CCC Read the next line of input text
CC
C
      CALL READF (IDCB, IERR, INTXT, IL)
      IF (IERR .LT. 0) GO TO 9000
C
CC
CCC Check to see it the line has "TOTAL..." in it, indicative of our EOF
CC
C
      IF (INTXT (2) .EQ. 2HOT) GO TO 200
C
CC
CCC While the word "FAULT..." does NOT appear, loop at 60 reading lines of
CC  input
C
         IF (INTXT (2)  NE. 2HAU) GO TO 60
C
CC
CCC Until it does..., then write out all subsequent lines, excluding any and
CC  all lines containing --- s until we get EOF or non blank first character,
C
70          WRITE (TAPENO,75)INTXT
75          FORMAT(18A2)
C
CC
CCC Increment record counter (to see if we need to write to second tape
CC
C
            COUNT = COUNT + 1
            IF (COUNT .GE. 1290) TAPENO = 5
```

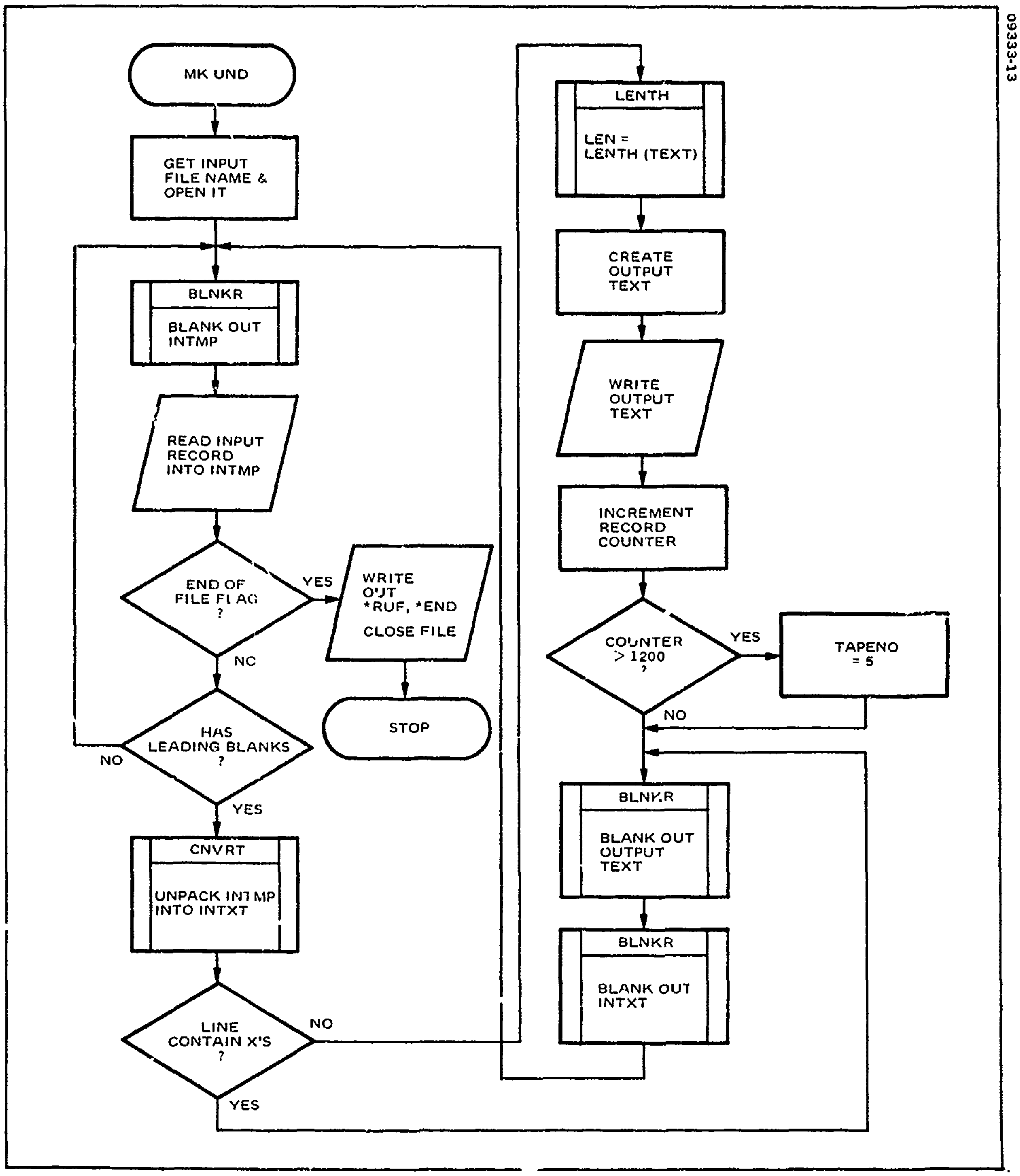
```
C
CC
CCC Blank out line of text in preparation to read the next one
CC
C
80          CALL BLNKR (INTXT)
C
CC
CCC Read the next line of input text
CC
C
            CALL READF (IDCB, IERR, INTXT, IL)
C
CC
CCC Check for EOF
CC
C
            IF (INTXT (2) .EQ. 2HOT) GO TO 200
C
CC
CCC If chars are any comb. of -,1,0 then skip this line
CC
C
               IF (INTXT (5) .LE. 020061B) GO TO 80
C
CC
CCC If the first few characters are non-blank, then we have reached the end
CC  of the current fault section, write no more lines, go back and read more
C
                  IF (INTXT (2) .NE. 2H  ) GO TO 60
C
CC
CCC Otherwise, go back and continue reading and writing...
CC
C
                     GO TO 70
200   CONTINUE
C
CC
CCC When we reach EOF. write out the EF for a flag
CC
C
      WRITE (TAPENO,205)EOF
205   FORMAT (A2)
      CALL CLOSE (IDCB)
      STOP
9000  WRITE (1,9010)
9010  FORMAT(" DATA FILE ERROR")
      STOP
      END
      SUBROUTINE BLNKR (INTXT)
C
CC
CCC This subroutine blanks out the text it is passed as input
CC
C
      IMPLICIT INTEGER (A-Z)
      DIMENSION INTXT (18)
      DO 10 I = 1, 18
         INTXT (I) = 2H
```

```
 10     CONTINUE
        RETURN
        END
        END$
```

SECTION A. 3

MKUND

A.3.1 – MKUND FLOW CHART



APPENDIX A – Software

Section A.3 – MKUND

A.3.2 – MKUND LISTING

```
MKUND  T=00004 IS ON CR00018 USING 00010 BLKS R=0000

FTN4.L
      PROGRAM MKUND
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C
C     PROGRAM:       MKUND
C
C     PURPOSE:       To take the faults detected output of LNDEL and make
C                    it into a +UNDected fault listing in a format which
C                    can be used as an input for further runs of SIMUL,
C                    enabling SIMUL to run faster and more efficiently.
C
C     PROGRAMMER:    DAVID S. WAGNER   -    BLDG. 688/T125
C                    7030 E POTAWATAMI      HUGHES/(213) 802-4190
C                    TUCSON, AZ 85715       FULLERTON, CA 92634
C                    602-296-2760
C
C     DATE:          23 - JUL - 80
C
C
C     DATA FILES:
C                    INPUT:    Any name desired, user is queried.  This file
C                              will be the output of the LNDEL program.
C
C                    OUTPUT:   Output will be directed to tape 4 for the
C                              first 1200 faults, thereafter to tape 5
C                              until 2400 are reached (the maximum)
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C SUBROUTINE   TYPE     DESCRIPTION
C ----------   ------   ----------------------------------------------
C
C BLNKP        User     This routine blanks out the array of text it's passed
C
C CODE         System   This routine reformats data in memory
C
C CNVRT        User     This subroutine is used to convert a packed line of
C                       text to an unpacked form for text processing
C
C LENTH        User     This function returns the lennth, in characters, of
C                       the array of text it is passed
C
C UNPAK        User     This routine unpacks one word in an A2 format into
C                       2 words each in an A1 format
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C ARRAYS USED IN THIS PROGRAM:
C
C ARRAY  (36)   Array used for blanking out purposes
C CHARS  (2)    Array into which an A2 word is unpacked
C END    (4)    Array holding "+END" characters for END of FILE
```

```
C NAME   (3)     Array to hold the input file name
C RUF    (4)     Array holding "*RUF" characters for Removing undetecable
C                faults prior to *END
C
C VARIABLES USED IN THIS PROGRAM:
C
C COUNT          Holds the count of number of records written to output tapes
C LINLEN         Holds the length in characters of the unpacked input record
C TAPENO         Holds the current tape drive no. for output
C TXTLEN         Holds the length in characters/2 (packed) of input records
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C GLOBAL INITIALIZATION:
C
      IMPLICIT INTEGER (A - Z)
      DIMENSION IDCB (144), INTMP (18), INTXT (36), OUTXT (36)
      DIMENSION CHARS (2), ARRAY (36), NAME (3), END (4), RUF (4)
      DATA END/1H*,1HE,1HN,1HD/
      DATA RUF/1H*,1HR,1HU,1HF/
      TXTLEN = 18
      LINLEN = 36
      COUNT = 0
      TAPENO = 4
C
CC
CCC Prompt to get the input file name from the user
CC
C
      WRITE (1,1)
1     FORMAT(" ENTER INPUT FILE NAME:")
      READ (1,2)NAME
2     FORMAT(3A2)
C
CC
CCC Open the input file name. if an open error. flag it and stop
CC
C
      CALL OPEN (IDCB, IERR. NAME)
      IF (IERR .LT. 0) GO TO 9000
C
CC
CCC Blank out the current line of text
CC
C
10    CALL BLNKP (INTMP, TXTLEN)
C
CC
CCC Read the next line of input text
CC
C
      CALL READF (IDCB  IERR. INTMP)
      IF (IERR .LT. 0) GO TO 9000
C
CC
CCC Check to see if the line has "EF" in it, indicative of EOF
CC
C
      IF (INTMP (1) .EQ. 2HEF) GO TO 120
```

```
C
CC
CCC Now, while the line doesn't start with blanks, repeat reading lines of
CC  input until it does (lines that don't start with blanks cant have faults
C
              IF (INTMP (1) .NE. 2H   ) GO TO 10
C
CC
CCC Now unpack the line so we can    i.t character positions more easily
CC
C
                CALL CNVRT (INTMP, INTXT, TXTLEN, LINLEN)
C
CC
CCC If the line has an X (or X s) discard it
CC
C
                IF (INTXT (10) .EQ. 1HX) GO TO 115
C
CC
CCC Get the length of input text in characters...
CC
C
                LEN = LENTH (INTXT)
C
CC
CCC Now create the output text record
CC
C
                OUTXT (1) = 1H*
                OUTXT (2) = 1HU
                OUTXT (3) = 1HN
                OUTXT (4) = 1HD
                OUTXT (5) = 1H
                DO 100 J = 6, (LEN - 14) + 7
                  OUTXT (J) = INTXT (J + 8)
100             CONTINUE
                OUTXT ( (LEN - 14) + 7) = 1H$
C
CC
CCC Write out the output text record to the current output tape drive
CC
C
                WRITE (TAPENO.110) OUTXT
110             FORMAT(36A1)
C
CC
CCC Increment record counter, and if greater than 1200, switch to drive 5
CC
C
                COUNT = COUNT + 1
                IF (COUNT .GE. 1200) TAPENO = 5
C
CC
CCC Now blank out input and output records preparatory to reading in another
CC
C
115             CALL BLNKR (OUTXT. LINLEN)
                CALL BLNKR (INTXT, LINLEN)
                GO TO 10
```

```
C
CC
CCC Now, write out the *RUF, and the *END at the end of the output file
CC
C
120   WRITE (TAPENO,125) RUF,END
125   FORMAT(4A1,/,4A1)
      CALL CLOSE (IDCB)
      STOP
9000  WRITE (1,9010)IERR
9010  FORMAT(" ERROR ON DATA FILE ERROR IS # ",I6)
      STOP
      END
      SUBROUTINE BLNKP (ARRAY, LENGTH)
C
CC
CCC This subroutine blanks out ARRAY dimensioned to LENGTH that it's passed
CC
C
      IMPLICIT INTEGER (A - Z)
      DIMENSION ARRAY (LENGTH)
      DO 10 I = 1, LENGTH
         IF (LENGTH .EQ. 18) ARRAY (I) = 2H
         IF (LENGTH .EQ. 36) ARRAY (I) = 1H
10    CONTINUE
      RETURN
      END
      FUNCTION LENTH (INTXT)
C
CC
CCC This function returns the number of characters in the string INTXT
CC
C
      IMPLICIT INTEGER (A - Z)
      DIMENSION INTXT (36)
      I = 36
5     IF (INTXT (I) .NE. 1H ) GO TO 20
         I = I - 1
         IF (I .NE. 1) GO TO 5
            LENTH = 1
            RETURN
20    LENTH = I
      RETURN
      END
      SUBROUTINE UNPAK (PAK, CHARS)
C
CC
CCC This subroutine Unpacks 2 characters in PAK (A2) into 2A1's in array CHAR
CC
C
      IMPLICIT INTEGER (A - Z)
      DIMENSION CHARS (2)
      CALL CODE
      READ (PAK, 5) CHARS
5     FORMAT(2A1)
      RETURN
      END
      SUBROUTINE CNVRT (INTMP, INTXT, TXTLEN, LINLEN)
C
CC
```

```
CCC This subroutine converts packed text in INTMP(TXTLEN) into unpacked
CC  text in INTXT(LINLEN)
C
      IMPLICIT INTEGER (A - Z)
      DIMENSION CHARS (2), TMP (2)
      DIMENSION INTMP (TXTLEN), INTXT (LINLEN)
      PTR = 0
      DO 10 I = 1, TXTLEN
         PTR = PTR + 1
         CALL UNPAK (INTMP (I), TMP)
         INTXT (PTR) = TMP (1)
         PTR = PTR + 1
         INTXT (PTR) = TMP (2)
10    CONTINUE
      RETURN
      END
      END$
```

SECTION A. 4

INIT/NOOP

A.4.1 – INIT LISTING

```
INIT   T=00004 IS ON CR00018 USING 00017 BLKS R=0000

FTN4,L
      PROGRAM INIT
      IMPLICIT INTEGER (A-Z)
      REAL TIME. VDH,VCH,VDL,VCL
      DIMENSION IDRIV4 (3). IDRIV7 (2), IDRIV8 (4), IDRTOG (2)
      DIMENSION IDRV12 (2), IHI7 (2), IHITOG (2), ILOTOG (2)
      DIMENSION IHI8 (2), IHI4 (2), ILO7 (2), ILO4 (2), ILO8 (3)
      DIMENSION ILO12 (2), IPINS (150)  IERR (4), IBUF (5). INAM (3)
      DATA IDRIV4/2,47.46/
      DATA IDRIV7/1,104/
      DATA IDRTOG/1,105/
      DATA IDRIV8/3,106,109,107/
      DATA IDRV12/1,177/
      DATA IHI4/1,46/
      DATA ILO4/1.47/
      DATA ILO7/1,104/
      DATA IHI8/1,106/
      DATA ILO8/3,109.107/
      DATA ILO12/1,177/
      DATA IHITOG/1,105/
      DATA ILOTOG/1,105/
      LDTU = 30
      MODE = 1
      ICODE = 23
      IBUF (1) = 2H:T
      IBUF (2) = 2HR,
      IBUF (3) = 2HPO
      IBUF (4) = 2HWO
      IBUF (5) = 2HH
      INAM (1) = 2HFM
      INAM (2) = 2HGR
      INAM (3) = 2H
      CALL EXEC (ICODE. INAM  0,0,0.0.0.IBUF.5)
      CALL XINIT (LDTU, IERR, MODE. IPINS)
      IF (IERR .NE. 0) GO TO 9000
      TIME = 1E-2
      VDH = 5E0
      VDL = 0E0
      VCH = 4E0
      VCL = 1E0
      NSET = 1
      CALL XTREF (LDTU. IERR. NSET. VDH, VDL. VCH, VCL)
      IF (IERR .NE. 0) GO TO 9000
      NCRD = 4
      CALL XWSET (LDTU, IERR, NSET, NCRD)
      IF (IERR .NE. 0) GO TO 9000
      CALL XTDRV (IERR. MODE. IDRIV4, IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XETHI (IERR. IHI4, IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XETLO (IERR, ILO4. IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XTEST (IERR. ISTAT, MODE, IPINS)
      IF (IERR .NE. 0) GO TO 9000
C       CALL PAUSR
      NCRD = 7
      CALL XWSET (LDTU. IERR. NSET. NCRD)
      IF (IERR .NE. 0) GO TO 9000
      CALL XTDRV (IERR, MODE, IDRIV7, IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XTDRV (IERR, MODE, IDRTOG, IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XETLO (IERR, ILO7, IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XETHI (IERR. IHITOG, IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XTEST (IERR, ISTAT, MODE, IPINS)
      IF (IERR .NE. 0) GO TO 9000
C       CALL PAUSR
      NCRD = 8
      CALL XWSET (LDTU, IERR, NSET, NCRD)
      IF (IERR .NE. 0) GO TO 9000
      CALL XTDRV (IERR, MODE. IDRIV8. IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XETHI (IERR, IHI8, IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XETLO (IERR, ILO8. IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XTEST (IERR, ISTAT, MODE, IPINS)
      IF (IERR .NE. 0) GO TO 9000
C       CALL PAUSR
      NCRD = 12
      CALL XWSET (LDTU, IERR. NSET, NCRD)
      IF (IERR .NE. 0) GO TO 9000
      CALL XTDRV (IERR, MODE, IDRV12, IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XETLO (IERR, ILO12, IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XTEST (IERR, ISTAT. MODE. IPINS)
      IF (IERR .NE. 0) GO TO 9000
C
CC
CCC NOW SET ADAPTER NO. 105 LOW
CC
C
C       CALL PAUSR
      NCRD = 7
      CALL XWSET (LDTU. IERR, NSET, NCRD)
      IF (IERR .NE. 0) GO TO 9000
      CALL XTDRV (IERR. MODE. IDRTOG, IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XETLO (IERR, ILOTOG, IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XTEST (IERR, ISTAT, MODE, IPINS)
      IF (IERR .NE. 0) GO TO 9000
C
CC
CCC NOW SET IT HIGH AGAIN
CC
C
C       CALL PAUSR
      CALL XETHI (IERR. IHITOG, IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XTEST (IERR, ISTAT, MODE, IPINS)
      IF (IERR .NE. 0) GO TO 9000
      STOP
9000  CONTINUE
      WRITE (1, 9010)IERR
```

```
9010  FORMAT(" IERR IS:",I2,1X,3A2)
      STOP
      END
      SUBROUTINE PAUSR
      IMPLICIT INTEGER (A-Z)
      WRITE (1,10)
10    FORMAT(" PAUSE <CR>")
      READ (1.20)IDUM
20    FORMAT(I3)
      RETURN
      END
      END$
```

A.4.2 – NOOP LISTING

```
NOOP   T=00004 IS ON CR00018 USING 00014 BLKS R=0000

FTN4,L
      PROGRAM NOOP
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C     PROGRAM:         NOOP
C
C     PURPOSE:         To initialize the hardware on the 1646178 board
C                      prior to testing.
C
C     PROGRAMMER:      DAVID S. WAGNER     - BLDG. 688/T125 (213) 802-4190
C
C     DATE:            JULY 2, 1980
C
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C SUBROUTINE  TYPE    DESCRIPTION
C ------------------------------------------------------------------------
C
C  EXEC       System  Used to execute a transfer filed prior to
C                     executing this program
C
C  XINIT      System  Used to initialize the pin programming array. See
C                     also DTS-70 Programmers Reference Manual pg 7-26
C  XDLY       System  Used to set Delay time interval for Driver/Comparato
C                     relays.  See DTS-70 P.R.M pg. 7-5
C  XTREF      System  Used to set the voltage reference levels for D/C
C                     See DTS-70 P.R.M. pg. 7-6
C  XWSET      System  Used to switch reference levels to D/C cards. See
C                     DTS-70 P.R.M. pg. 7-7
C  XTDRV      System  Used to enable the Driver on a specified list of pins
C                     See P.R.M. pg. 7-12
C  XTCMP      System  Used to enable the Comparator on specified pin list
C                     See P.R.M. pg. 7-13
C  XETHI      System  Used to define pins desired to be set hi. See DTS-70
C                     P.R.M. pg. 7-14
C  XETLO      System  Used to define pins desired to be set lo. See DTS-70
C                     P.R.M. pg. 7-15
C  XTEST      System  Used to actually do the digital test based on pin
C                     state array  See P.R.M. pg. 7-16
C  XSERN      System  Used to report any and all subroutine errors. See
C                     DTS-70  Prog. Ref. Man
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C ARRAYS USED IN THIS PROGRAM:
C
C IDRIV4  (10)    Array to hold pin number(s) on D/C card no. 4
C IDRIV5  (8)     Array to hold pin number(s) on D/C card no. 5
C IERR    (4)     Error parameter holding array
C IDRIV6  (9)     Array to hold pin number(s) on D/C card no. 6
C IDRIV7  (4)     Array to hold pin number(s) on D/C card no. 7
C IDRIV8  (4)     Array to hold pin number(s) on D/C card no. 8
C IDRIVT  (2)     Array to hold pin number that is to be toggled
C IPINS   (150)   Pin state array needed to do digital tests
C IHI4    (4)     Array to hold pin numbers on D/C card 4 to set hi
```

```
C IHI5     (7)     Array to hold pin numbers on D/C card 5 to set hi
C ILO5     (2)     Array to ho d pin number on D/C card 5 to set lo
C IHI6     (9)     Array to hold pin numbers on D/C card 6 to set hi
C IHI7     (3)     Array to hold pin numbers on D/C card 7 to set hi
C ILO7     (2)     Array to hold pin numbers on D/C card 7 to set lo
C IHI8     (3)     Array to hold pin numbers on D/C card 8 to set hi
C ILO8     (2)     Array to hold pin numbers on D/C card 8 to set lo
C IHITOG   (2)     Array to hold pin number to toggle hi
C ILOTOG   (2)     Array to hold pin number to toggle lo
C IBUF     (5)     Array holds name of transfer file to execute
C INAM     (3)     Array holds name of system prog. to execute (FMGR)
C
C
C VARIABLES USED IN THIS PROGRAM:
C
C LDTU             Holds the logical unit no. for the Digital Test Unit
C MODE             Defines the mode of current test (See DTS-70 PRM sec. 7)
C NCRD             Holds the D/C card no. of card currently being set
C NSET             Holds the reference set number of current set
C TIME             Holds the D/C relay delay time in milliseconds
C VCH              Defines the voltage comparator High
C VCL              Defines the voltage comparator Lo
C VDH              Defines the voltage driver High
C VDL              Defines the voltage driver Lo
C
C GLOBAL INITIALIZATION:
C
C
      DIMENSION IDRIV4 (10), IDRIV5 (8), IDRIV6 (9), IDRIV7 (4)
      DIMENSION IDRIV8 (4), IDRIVT (2), IHITOG (2), ILOTOG (2)
      DIMENSION IPINS (150), IERR (4), IHI4 (2), IHI5 (7), IHI6 (9)
      DIMENSION IHI7 (3), IHI8 (3), ILO4 (9), ILO5 (2), ILO8 (2)
      DIMENSION IBUF (5), INAM (3), ILO7 (2)
      DATA IDRIV4/9,46,47,48,50,52,54,56,58,60/
      DATA IDRIV5/7,64,66,68,70,72,74,62/
      DATA IDRIV6/8,76,78,80,82,84,86,88,90/
      DATA IDRIV7/3,92,94,104/
      DATA IDRIV8/3,106,107,108/
      DATA IDRIVT/1,105/
      DATA IHITOG/1,105/
      DATA ILOTOG/1,105/
      DATA IHI4/1,46/
      DATA IHI5/6,64,66,68,70,72,74/
      DATA IHI6/8,76,78,80,82,84,86,88,90/
      DATA IHI7/2,92,94/
      DATA IHI8/2,106,108/
      DATA ILO4/8,47,48,50,52,54,56,58,60/
      DATA ILO5/1,62/
      DATA ILO7/1,104/
      DATA ILO8/1,107/
      LDTU = 30
      MODE = 1
      ICODE = 23
C
CC
CCC The following code executes FMGR which runs the transfer file POWON
CC
C
      IBUF (1) = 2H:T
      IBUF (2) = 2HR,
   IBUF (3) = 2HPO
   IBUF (4) = 2HWO
   IBUF (5) = 2HN
   INAM (1) = 2HFM
   INAM (2) = 2HGR
   INAM (3) = 2H
   CALL EXEC (ICODE,INAM,0,0,0,0,0,IBUF,5)
C
CC
CCC Now initialize the DTU software
CC
C
   CALL XINIT (LDTU, IERR, MODE, IPINS)
   IF (IERR .NE. 0) GO TO 9000
C
CC
CCC Now define our delay time as 10 mSec
CC
C
   TIME = 1E-2
   CALL XDLY (LDTU, IERR, TIME)
   IF (IERR .NE. 0) GO TO 9000
C
CC
CCC Now initialize our voltage references
CC
C
   VDH = 5E0
   VDL = 0E0
   VCH = 4E0
   VCL = 1E0
   NSET = 1
   CALL XTREF (LDTU, IERR, NSET, VDH, VDL, VCH, VCL)
   IF (IERR .NE. 0) GO TO 9000
   NCRD = 4
C
CC
CCC And then switch them to our current reference set
CC
C
   CALL XWSET (LDTU, IERR, NSET. NCRD)
   IF (IERR .NE  0) GO TO 9000
C
CC
CCC Now enable the driver on card 4
CC
C
   CALL XTDRV (IERR, MODE, IDRIV4, IPINS)
   IF (IERR .NE. 0) GO TO 9000
C
CC
CCC and set the pins on that card to their proper states
CC
C
   CALL XETHI (IERR, IHI4, IPINS)
   IF (IERR .NE. 0) GO TO 9000
   CALL XETLO (IERR. ILO4, IPINS)
   IF (IERR .NE. 0) GO TO 9000
C
CC
```

```
CCC and now perform the actual setting of those pins
CC
C
      CALL XTEST (IERR, ISTAT, MODE, IPINS)
      IF (IERR .NE. 0) GO TO 9000
C
CC
CCC now switch to card no. 7 and repeat the process
CC
C
      NCRD = 7
      CALL XTREF (LDTU, IERR, NSET, VDH, VDL, VCH, VCL)
      IF (IERR .NE. 0) GO TO 9000
      CALL XWSET (LDTU, IERR, NSET, NCRD)
      IF (IERR .NE. 0) GO TO 9000
      CALL XTDRV (IERR, MODE, IDRIV7, IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XETLO (IERR, ILOTOG, IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XTEST (IERR, ISTAT, MODE, IPINS)
      IF (IERR .NE. 0) GO TO 9000
C
CC
CCC now card 5 is up to bat ..
CC
C
      NCRD = 5
      CALL XTREF (LDTU, IERR, NSET, VDH, VDL, VCH, VCL)
      IF (IERR .NE. 0) GO TO 9000
      CALL XWSET (LDTU, IERR, NSET, NCRD)
      IF (IERR .NE. 0) GO TO 9000
      CALL XTDRV (IERR, MODE, IDRIV5, IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XETHI (IERR, IHI5, IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XETLO (IERR, ILO5, IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XTEST (IERR, ISTAT, MODE, IPINS)
      IF (IERR .NE. 0) GO TO 9000
C
CC
CCC followed by card 6
CC
C
      NCRD = 6
      CALL XTREF (LDTU, IERR, NSET, VDH, VDL, VCH, VCL)
      IF (IERR .NE. 0) GO TO 9000
      CALL XWSET (LDTU, IERR, NSET, NCRD)
      IF (IERR .NE. 0) GO TO 9000
      CALL XTDRV (IERR, MODE, IDRIV6, IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XETHI (IERR, IHI6, IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XTEST (IERR, ISTAT, MODE, IPINS)
      IF (IERR .NE. 0) GO TO 9000
C
CC
CCC then it is card 7's turn again
CC
C
```

```
      NCRD = 7
      CALL XTREF (LDTU, IERR, NSET, VDH, VDL, VCH, VCL)
      IF (IERR .NE. 0) GO TO 9000
      CALL XWSET (LDTU, IERR, NSET, NCRD)
      IF (IERR .NE. 0) GO TO 9000
      CALL XTDRV (IERR, MODE, IDRIV7, IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XETHI (IERR, IHI7, IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XTEST (IERR, ISTAT, MODE, IPINS)
      IF (IERR .NE. 0) GO TO 9000
C
CC
CCC then 8 comes...
CC
C
      NCRD = 8
      CALL XTREF (LDTU, IERR, NSET, VDH, VDL, VCH, VCL)
      IF (IERR .NE. 0) GO TO 9000
      CALL XWSET (LDTU, IERR, NSET, NCRD)
      IF (IERR .NE. 0) GO TO 9000
      CALL XTDRV (IERR, MODE, IDRIV8, IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XETHI (IERR, IHI8, IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XETLO (IERR, ILO8, IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XTEST (IERR, ISTAT, MODE, IPINS)
      IF (IERR .NE. 0) GO TO 9000
C
CC
CCC NOW TOGGLE PIN 105
CC
C
      NCRD = 7
      CALL XTREF (LDTU, IERR, NSET, VDH, VDL, VCH, VCL)
      IF (IERR .NE. 0) GO TO 9000
      CALL XWSET (LDTU, IERR, NSET, NCRD)
      IF (IERR .NE. 0) GO TO 9000
      CALL XTDRV (IERR, MODE, IDRIVT, IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XETLO (IERR, ILOTOG, IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XTEST (IERR, ISTAT, MODE, IPINS)
      IF (IERR .NE. 0) GO TO 9000
C
CC
CCC NOW SET IT HIGH AGAIN
CC
C
      CALL XTREF (LDTU  IERR, NSET, VDH, VDL, VCH, VCL)
      IF (IERR .NE. 0) GO TO 9000
      CALL XWSET (LDTU, IERR, NSET, NCRD)
      IF (IERR .NE. 0) GO TO 9000
      CALL XTDRV (IERR  MODE, IDRIVT, IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XETHI (IERR, IHITOG, IPINS)
      IF (IERR .NE. 0) GO TO 9000
      CALL XTEST (IERR, ISTAT, MODE, IPINS)
      IF (IERR .NE. 0) GO TO 9000
```

```
C     WRITE (1,10)
10    FORMAT(" FINISHED TOGGLING PIN 105",/," PAUSING HERE UNTIL <CR>")
C     READ (1,20)IDUMMY
20    FORMAT(I3)
      STOP
9000  CONTINUE
C
CC
CCC THIS SECTION HANDLES ERRORS ON DT570 SUBROUTINE CALLS...
CC
C
      WRITE (1, 9010)IERR
9010  FORMAT(" IERR IS: ",I2,1X,3A2)
      CALL XSERN (LDTU, IERR(1))
      STOP
      END
      END$
```

SECTION A.5

DFISML

APPENDIX A – Software
Section A. 5 – DFISML

A.5.1 – SUPPORT MAINTENANCE SCHEMA

```
DFI         DTS-70        SUPPORT MAINTENANCE
            SCHEMA
$ CONTROL:TABLE,FIELD;
  Begin Data Base:DFISML:10:19;       ((Image Data Base NAMR))
     ((Security Information))
LEVELS:
     1     Leader                     ((Lowest Level and Code Pass Word))
     3     Senior
     5     Admin                      (( Highest Level and Code Pass Word))
     (( Item Definition))
ITEMS:
     PNUMB, x12 (1,3);                ((Part Number))
     SANUM, x12 (1,3);                ((Serial or Assembly Number))
     SACTV, x20 (1,3);                ((Support Activity))
                                      ((Software-Update; Generation))
                                      ((Hardware-Replace; Repair))
     STIME, R2 (1,3);                 (( Support Start Time))
     FTIME, R2 (1,3);                 (( Support Finish Time))
     ELTIME, R2 (3,5);                ((Support Elapsed Time))
     LCHARG, x10 (3,5);               ((Labor $ Charge))
     MCHARG, x10 (3,5);               ((Material $ Charge))
     DATE    , x12 (1,3);             ((Support-Maintenance Date))
   ((Set Definition))
SETS:
NAME: PART: : 19, A; ((Part Number File,Automatic Master))
ENTRY: PNUMB  (1);
CAPACITY: 23;
NAME: DATEF: : 19, A;                 ((Service Date File, Automatic Master))
ENTRY: DATE (1);
CAPACITY: 67 ;
NAME: SMFILE : : 19, D;               ((Support-Maint. Detail File))
ENTRY: PNUMB (PART), SANUM, SACTV, STIME, FTIME, ELTIME, LCHARG,
          MCHARG, DATE (DATEF);
CAPACITY: 161 ;
END.
```

A..5.2 – SUPPORT MAINTENANCE REPORT SMRPT

```
DFI       DTS-70          SUPPORT MAINTENANCE
QUERY-REPORT
          DFISML:10:19,ADMIN;
          Report NAME = SMRPT         (( SM Report Print Procedure File))
          H1, "DFI Support Maintenance Report", 81;
          H1, "Page", 107;
          H1, Page No, 111;
          H2, "DTS-70 System", 73;
          H3, "HAC---- Org-12-42-50----", 82, SPACE A2, E1;
          H4, "---- Part ------------ Serial/Asmby -------, Support-Activity-----Suppo rt
               ----Labor--------------Material---------Date---". 119;
          H5, "----Number--------------Number-----------------Time-Hr.
               ----$------------------$----------------", 119, Space A2
          S2, Date;
          S1, PNUMB;
          D1, Part, 26:
          D1, SANUM, 42;
          D1, SACTV, 66;
          D1, ELTIME, 79;
          D1, LCHARG, 93;
          D1, MCHARG, 107;
          D1, DATE, 119;
          G2, DATE, 26, E1;
          G1, PNUMB, 26;
          T2, "Date _ Sub-Totals", 66;
          T2, ELTIME, 79; Add;
          T2, LCHARG, 93, Add;
          T2, MCHARG, 107, Space B2, Add;
          TF, "Report Totals", 66;
          TF, ELTIME, 79, Add;
          TF, LCHARG, 93, Add;
          TF, MCHARG, 107, Space B2, Add;
          E1, "XX/XX/XX";
          END; "XX_XXX_XXXX_";
```

APPENDIX B – SCHEMATICS

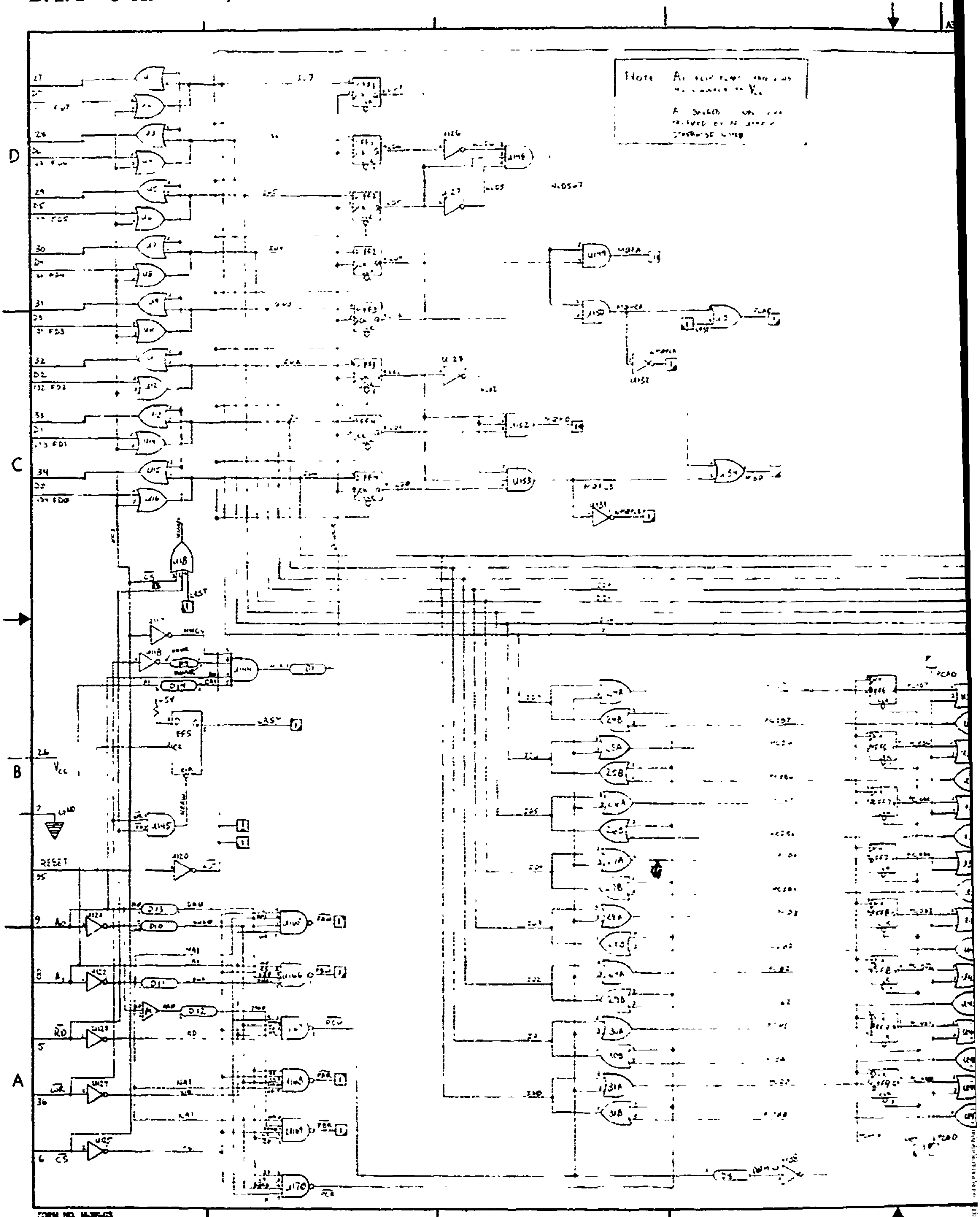
SECTION B.1

PN 1635972 CIRCUIT BOARD MODEL

APPENDIX B – Schematics

Section B. 1 – PN 1635972 Circuit Board Model

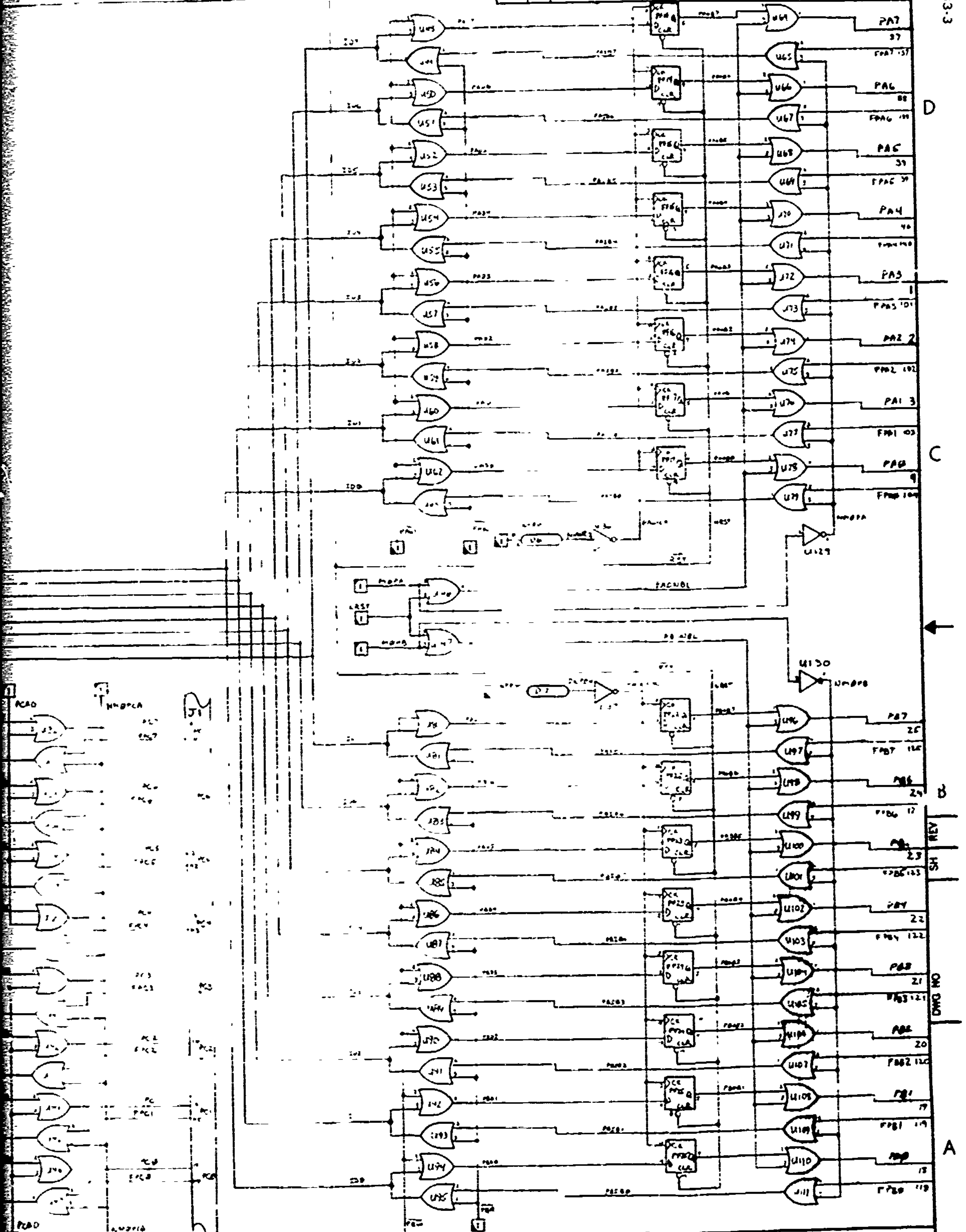
B. 1. 1 – 8 CHANNEL, 4 PORT PROGRAMMABLE PERIPHERAL INTERFACE, 8255



FORM NO. 16-385-G3
DIETERICH-POST CLEARPRINT 1000H-10

REVISIONS			2/20/80	3/7/80
ZONE	LTR	INTEL 8255 DESCRIPTION MODEL	DATE	APPROVED

09333-3

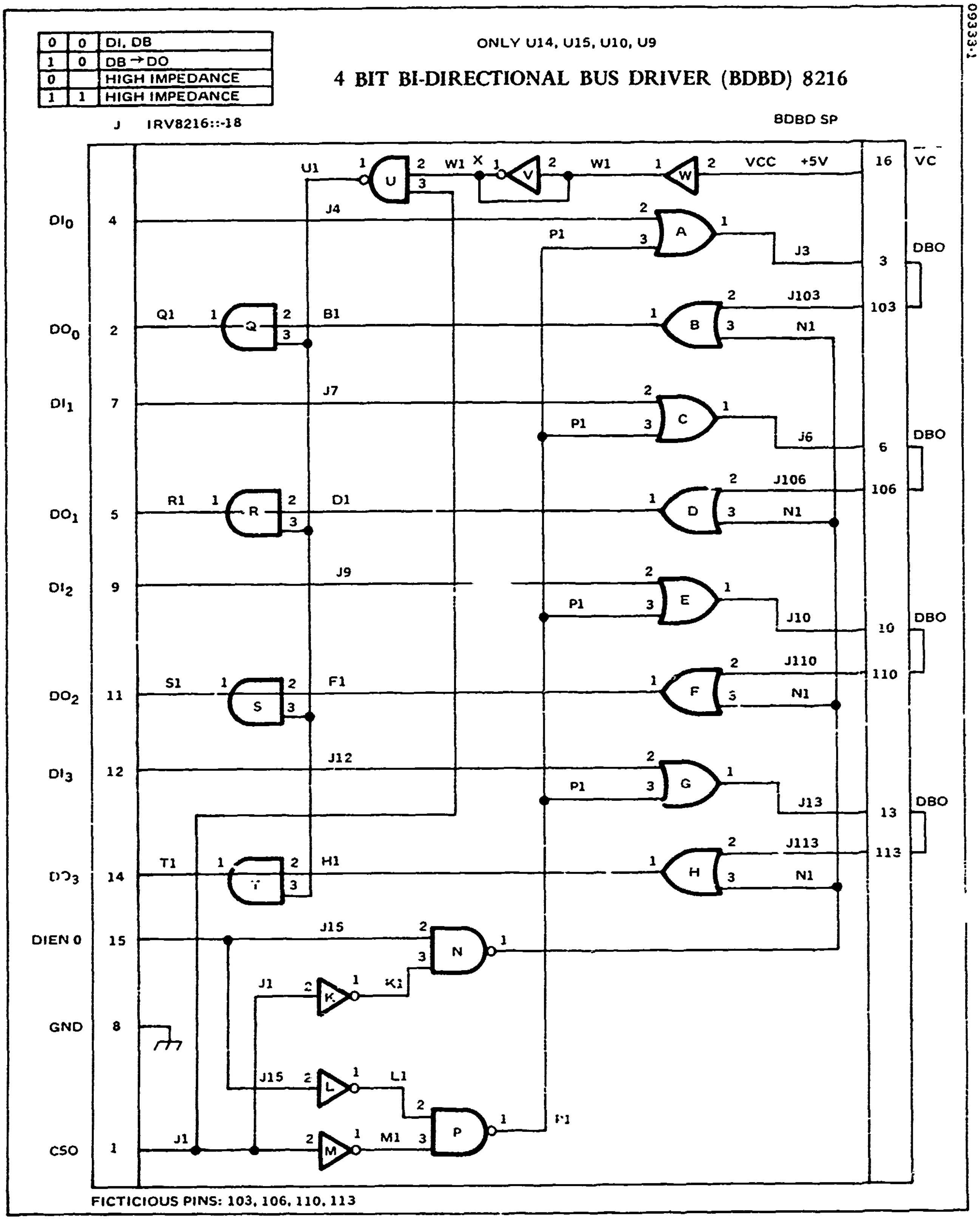


INTEL 8255 Peripheral Processor	CONTRACT: DFI-MMT	July 10, 1980		
DR	SIZE D	FSCM NO Rev. C	DWG NO.	DESIGN PAUL CHEY DAVID WAGNER
	SCALE None	MS 607/J3-5	2 6763	SHEET 1 of 1

2

APPENDIX B – Schematics
Section B. 1 – PN 1635972 Circuit Board Model

B.1.2 – 4 BIT BI-DIRECTIONAL BUS DRIVER, 8216



0	0	DI, DB
1	0	DB → DO
0		HIGH IMPEDANCE
1	1	HIGH IMPEDANCE

B.1.3 – 4 BIT BI-DIRECTIONAL BUS DRIVER, 8216A

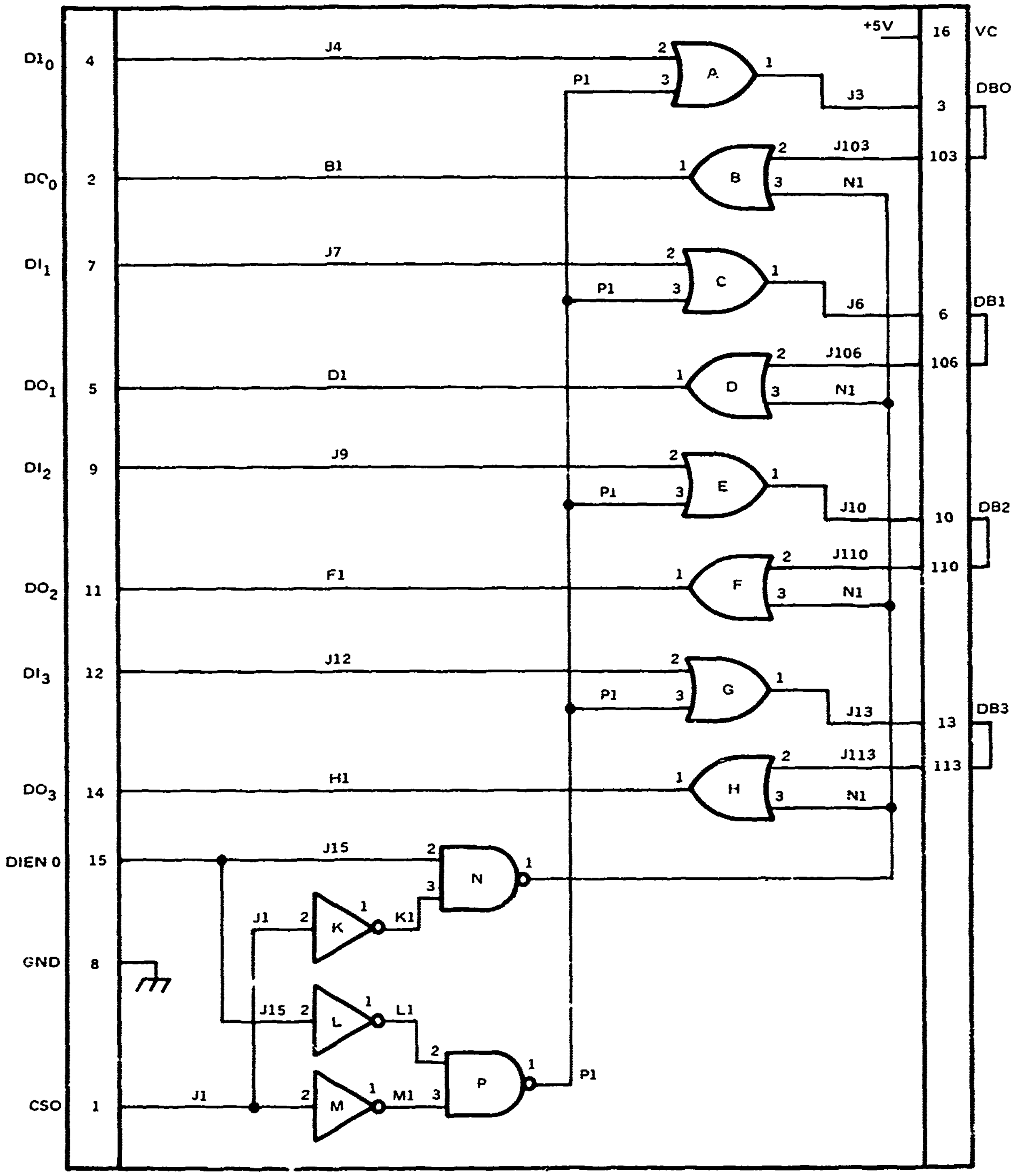
09333-25

0	0	DI, DB
1	0	DB → DO
0		HIGH IMPEDANCE
1	1	HIGH IMPEDANCE

J IRV8216::-18

ONLY U14, U15, &10, U9

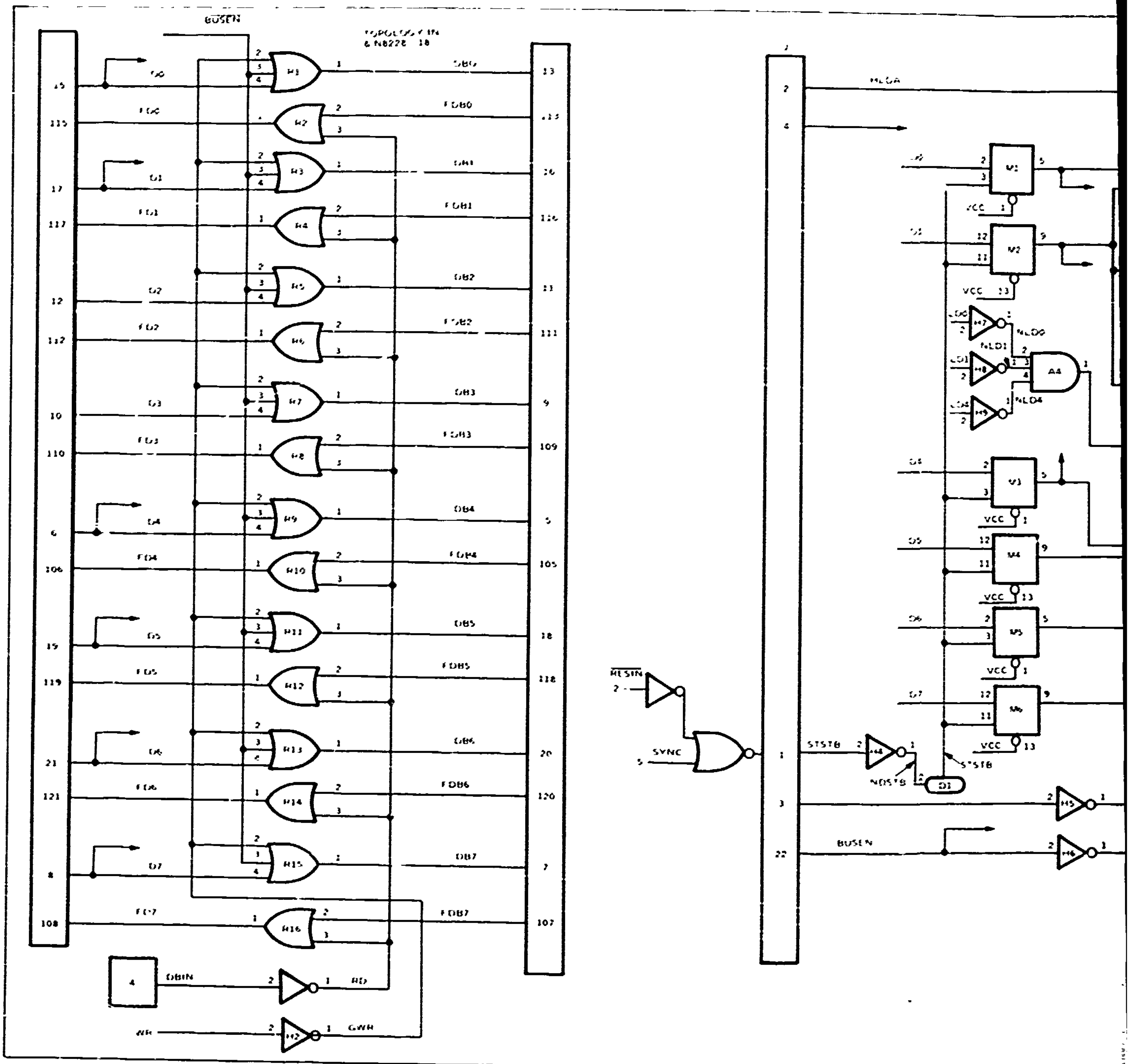
4 BIT BI-Directional Bus Driver (BDBD) 8216A

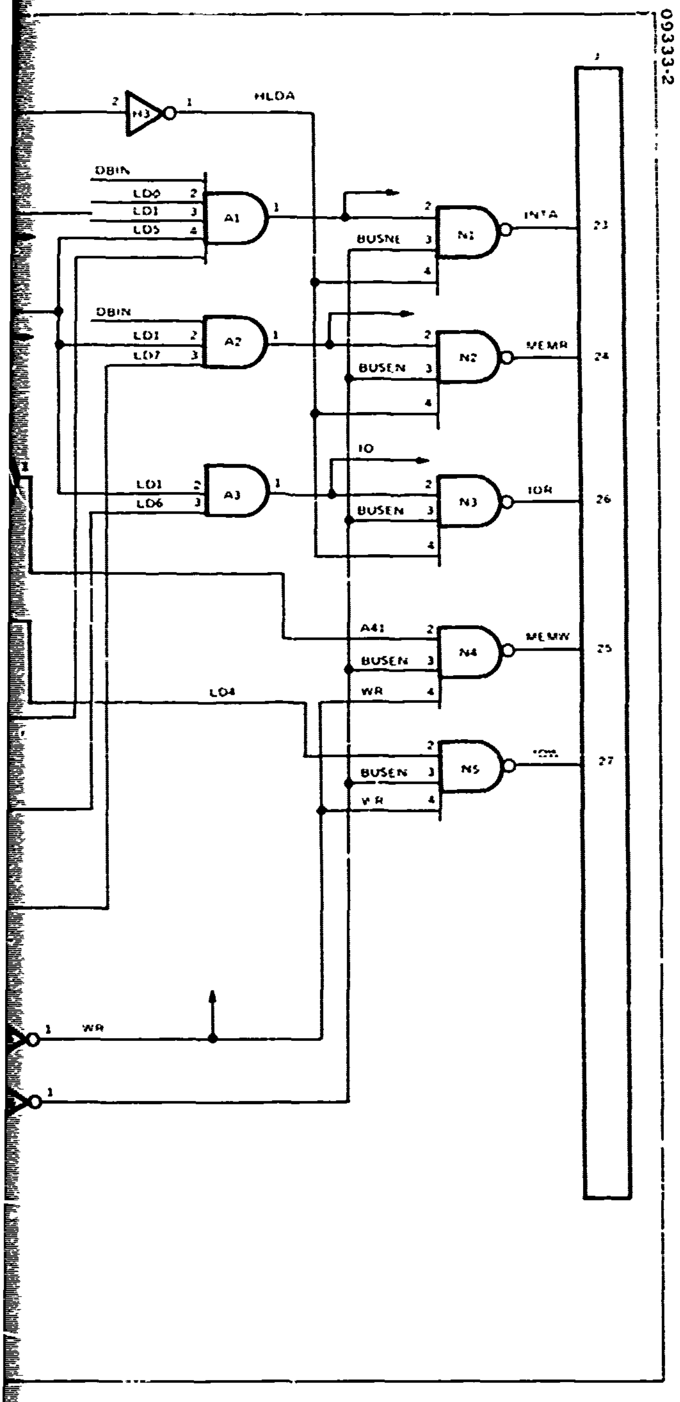


FICTICIOUS PINS: 103, 106, 110, 113

APPENDIX B – Schematics
Section B. 1 – PN 1635972 Circuit Board Model

B. 1.4 – SYSTEM CONTROLLER AND BUS DRIVER, 8228



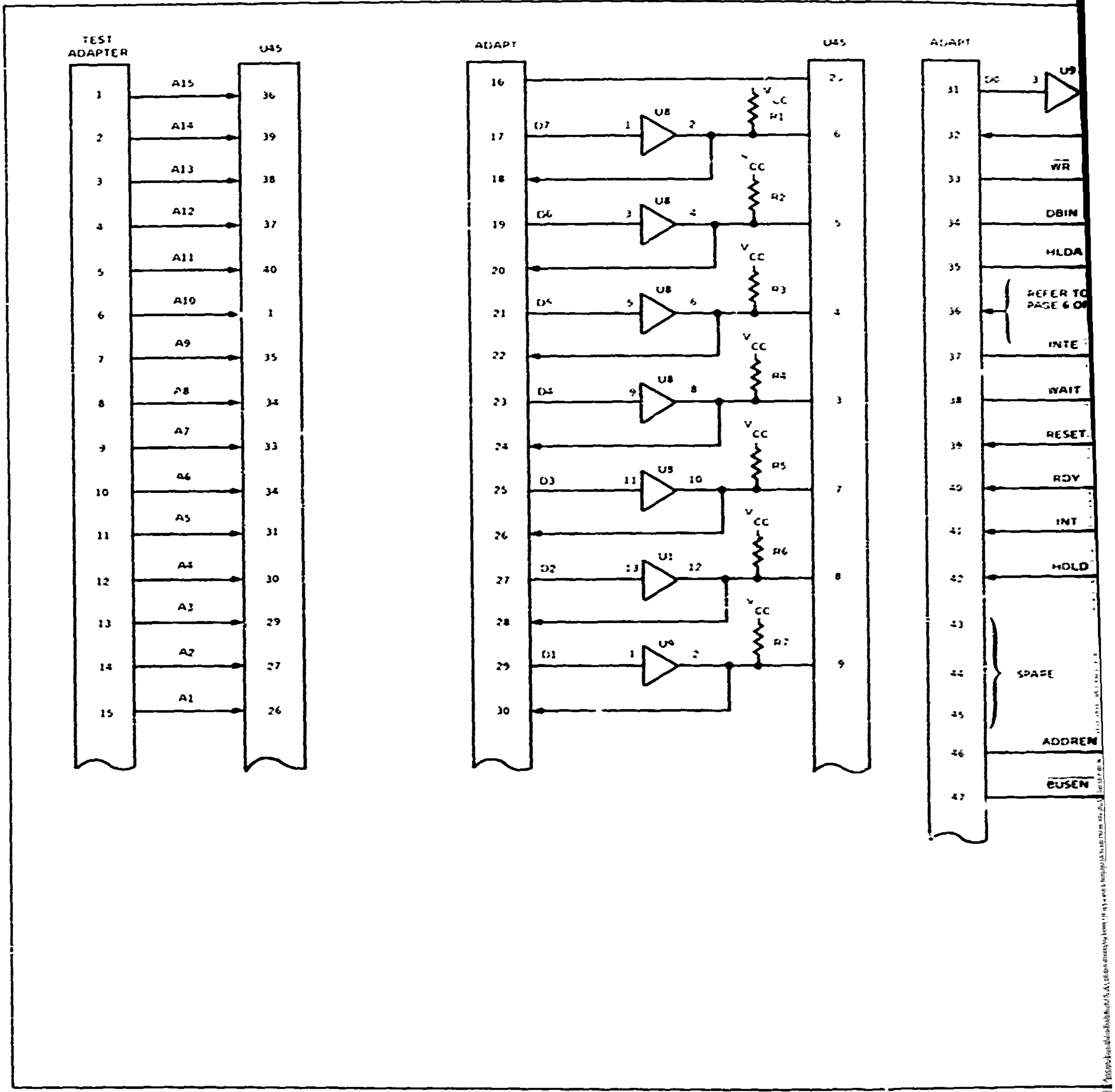

09333-2
HLDA
N3
DBIN
LD0
LD1
LD5
A1
INTA
BUSNE
N1
DBIN
LD1
LD7
A2
MEMR
BUSEN
N2
IO
LD1
LD6
A3
IOR
BUSEN
N3
A41
BUSEN
WR
N4
MEMW
LD4
BUSEN
WR
N5
IOW
WR
J
23
24
26
25
27

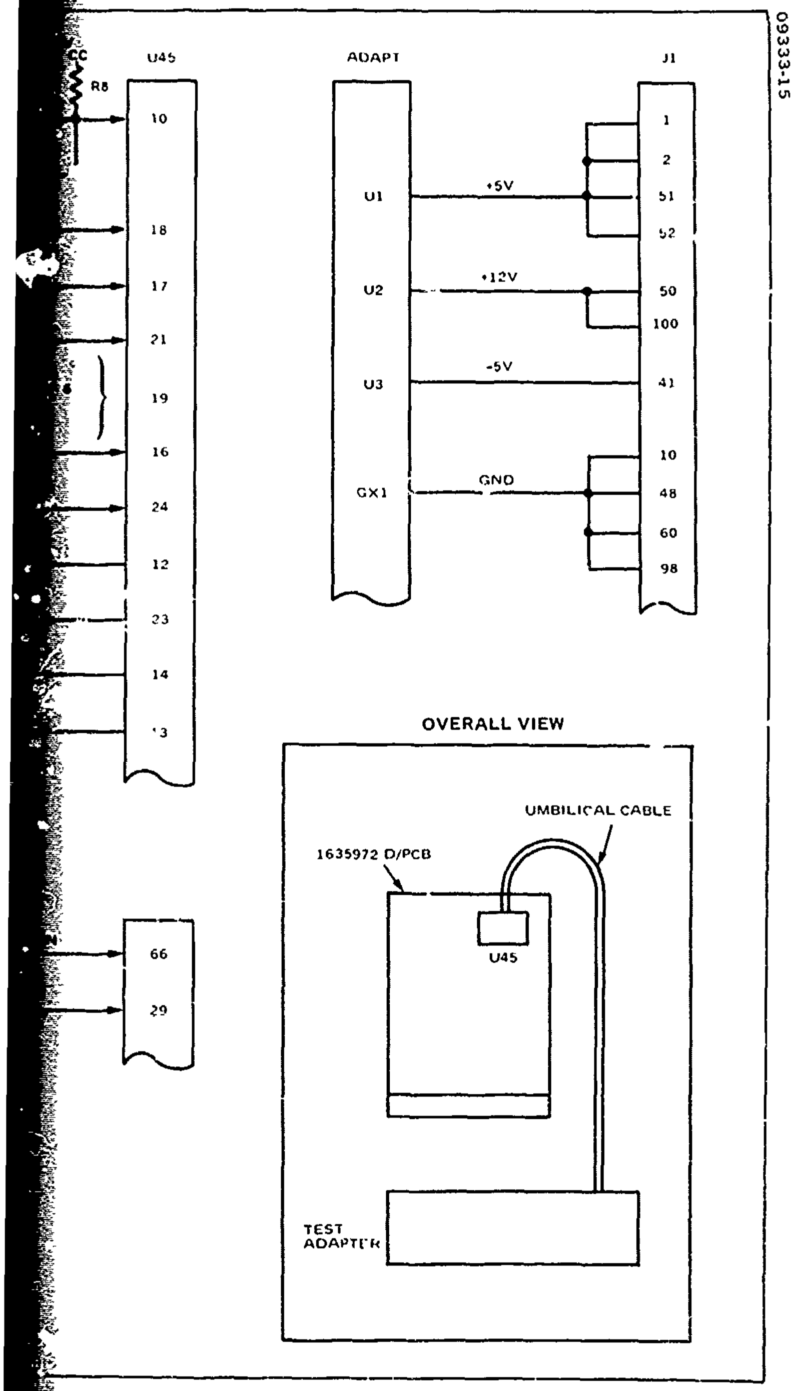
2

1

APPENDIX B – Schematics
Section B. 2 – PN 1635972 Circuit Board Test Adapter

B. 2. 1 – 8080 A/B UMBILICAL CABLE; PART 1 OF 6

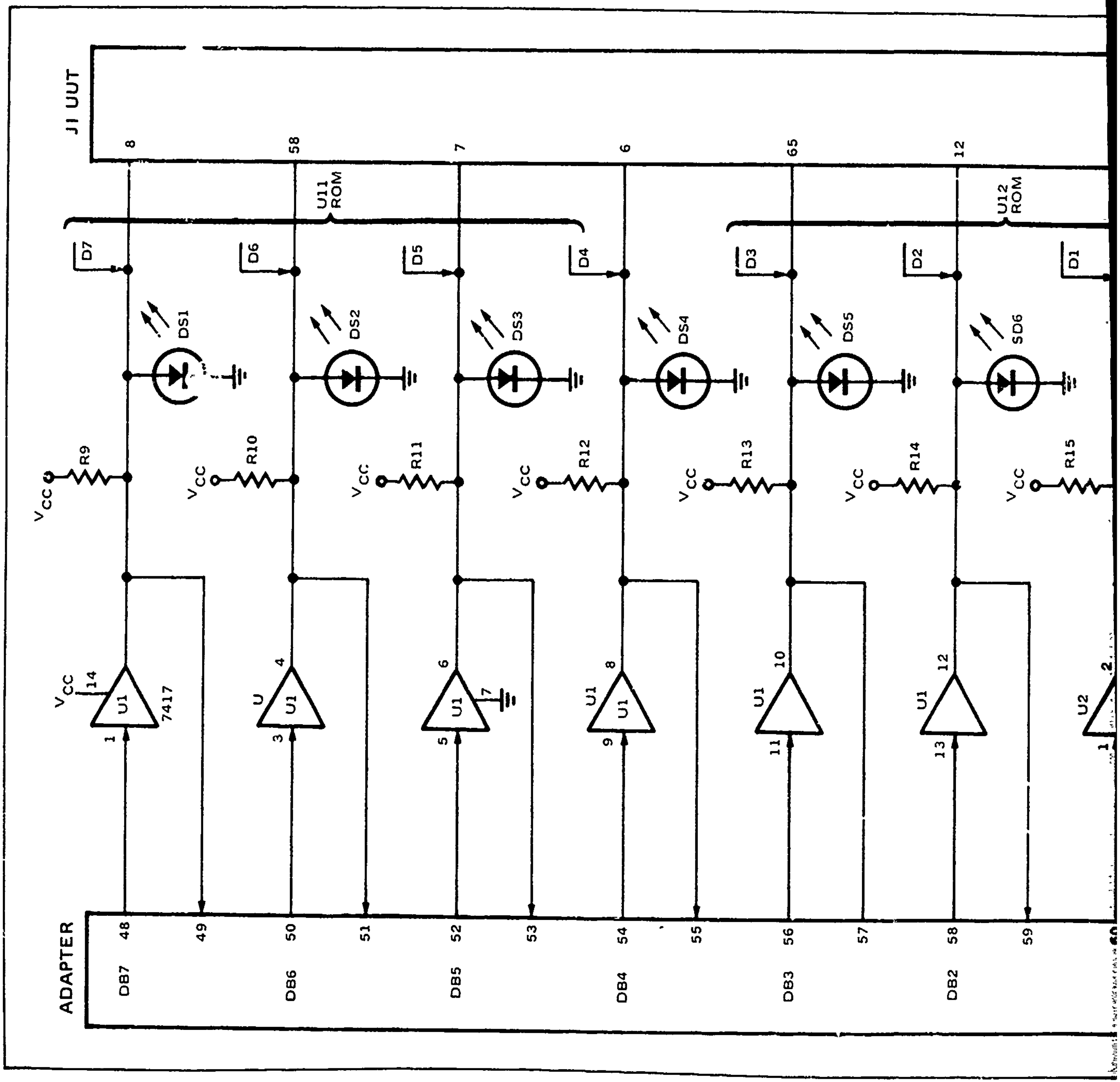


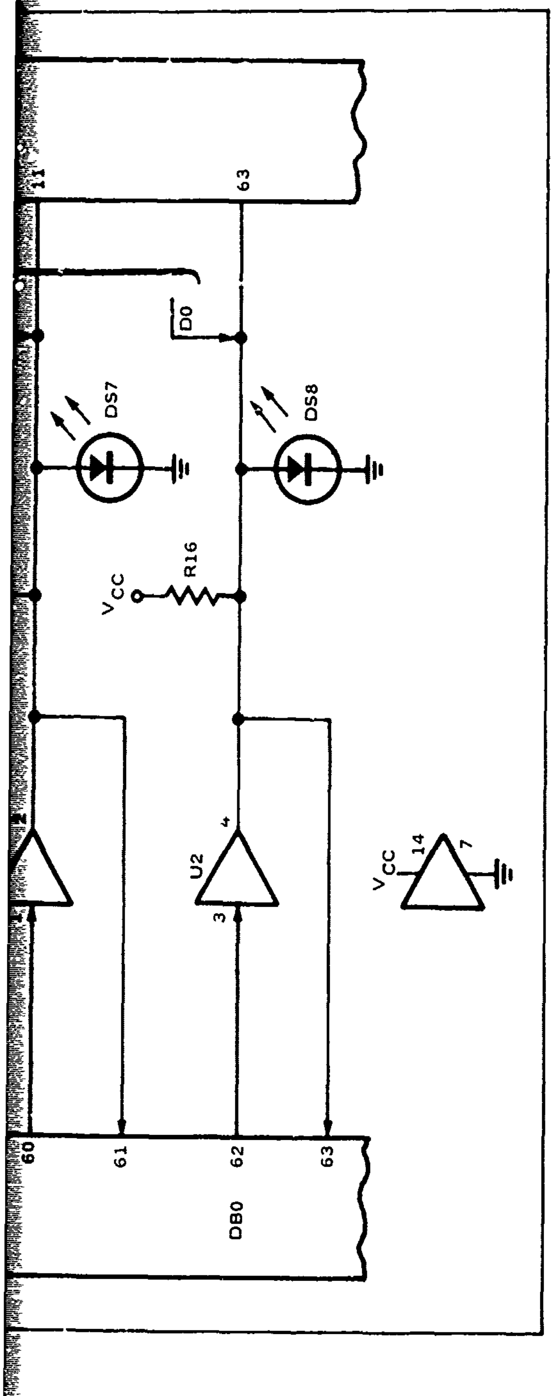

09333-15
CC
R8
U45
10
18
17
21
19
16
24
12
23
14
13
66
29
ADAPT
U1
U2
U3
GX1
+5V
+12V
-5V
GND
J1
1
2
51
52
50
100
41
10
48
60
98
OVERALL VIEW
UMBILICAL CABLE
1635972 D/PCB
U45
TEST ADAPTER

APPENDIX B – Schematics
Section B.2 – PN1635972 Circuit Board Test Adapter

B.2.2 – 8 Bit Data Bus, Buffers, Pull Up and LED Indicators; Part 2 of 6

09333-16

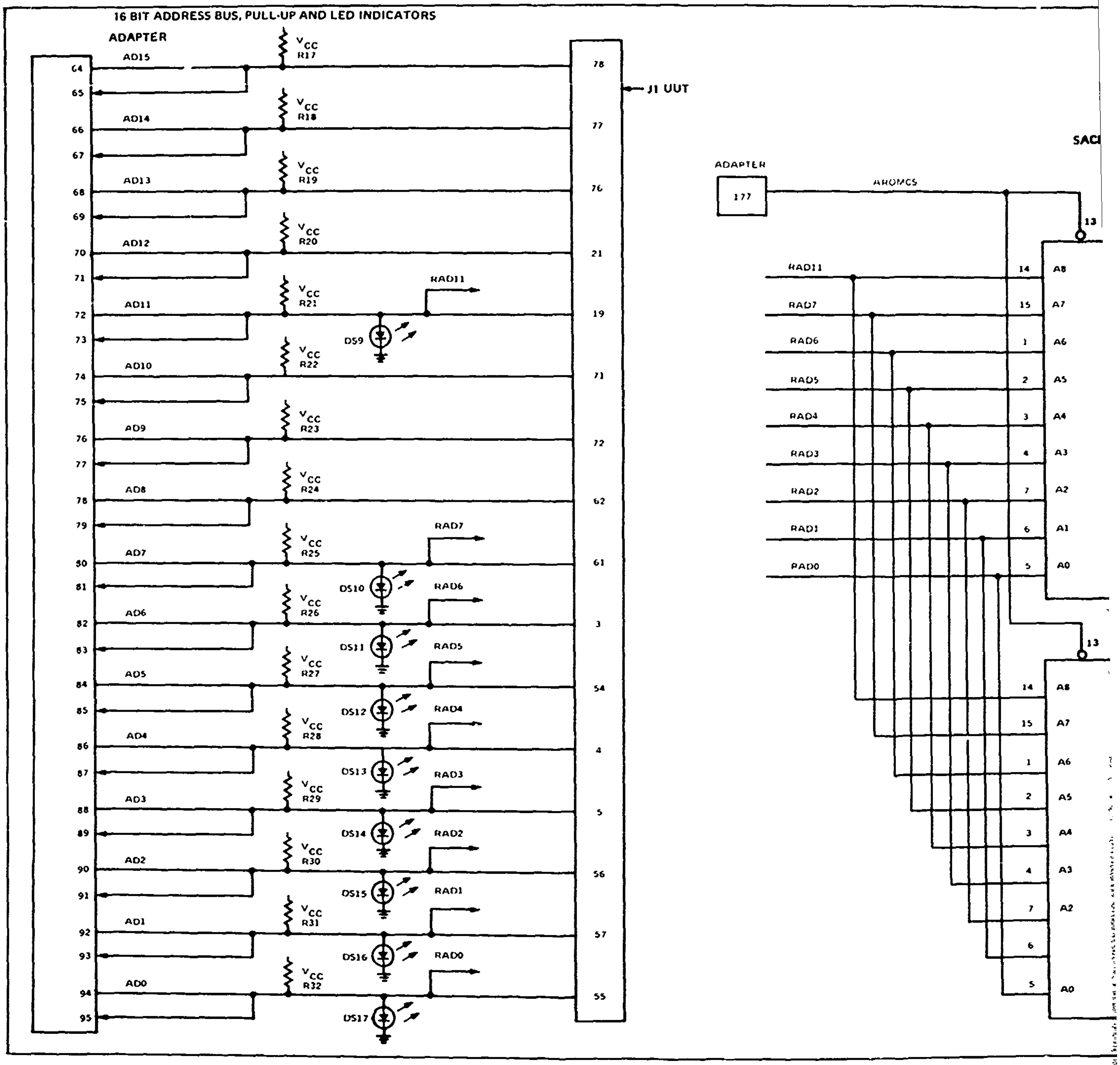


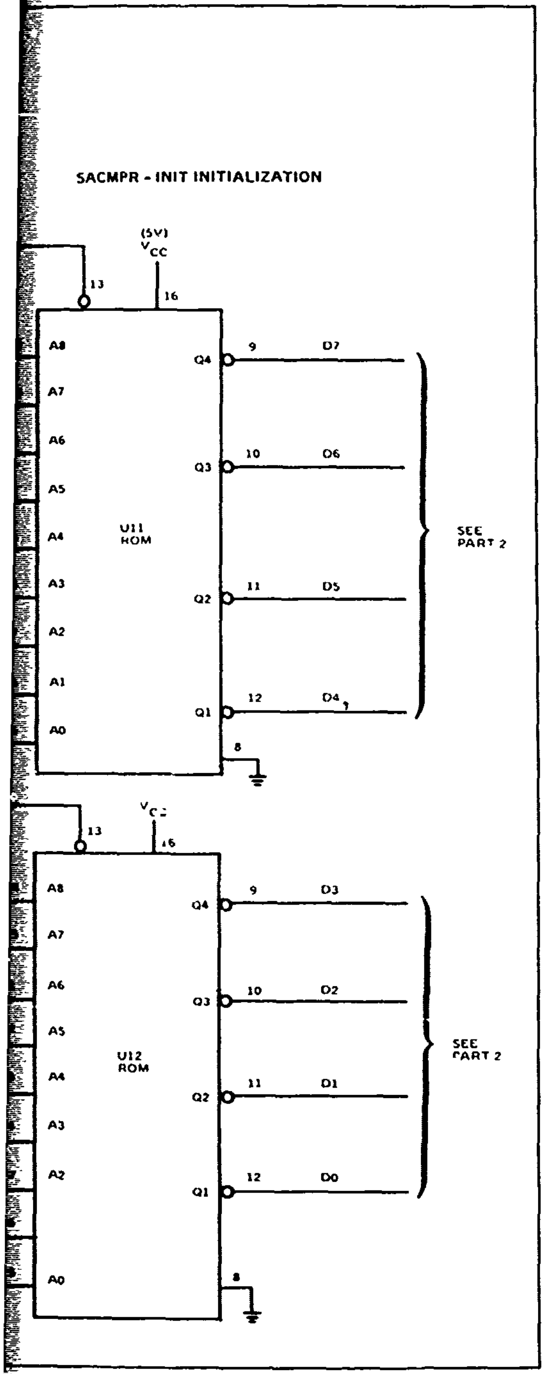


2

APPENDIX B – Schematics
Section B.2 – PN 1635972 Circuit Board Test Adapter

B.2.3 – 16 Bit Address Bus, SACMPR-INIT Initialization; Part 3 of 6





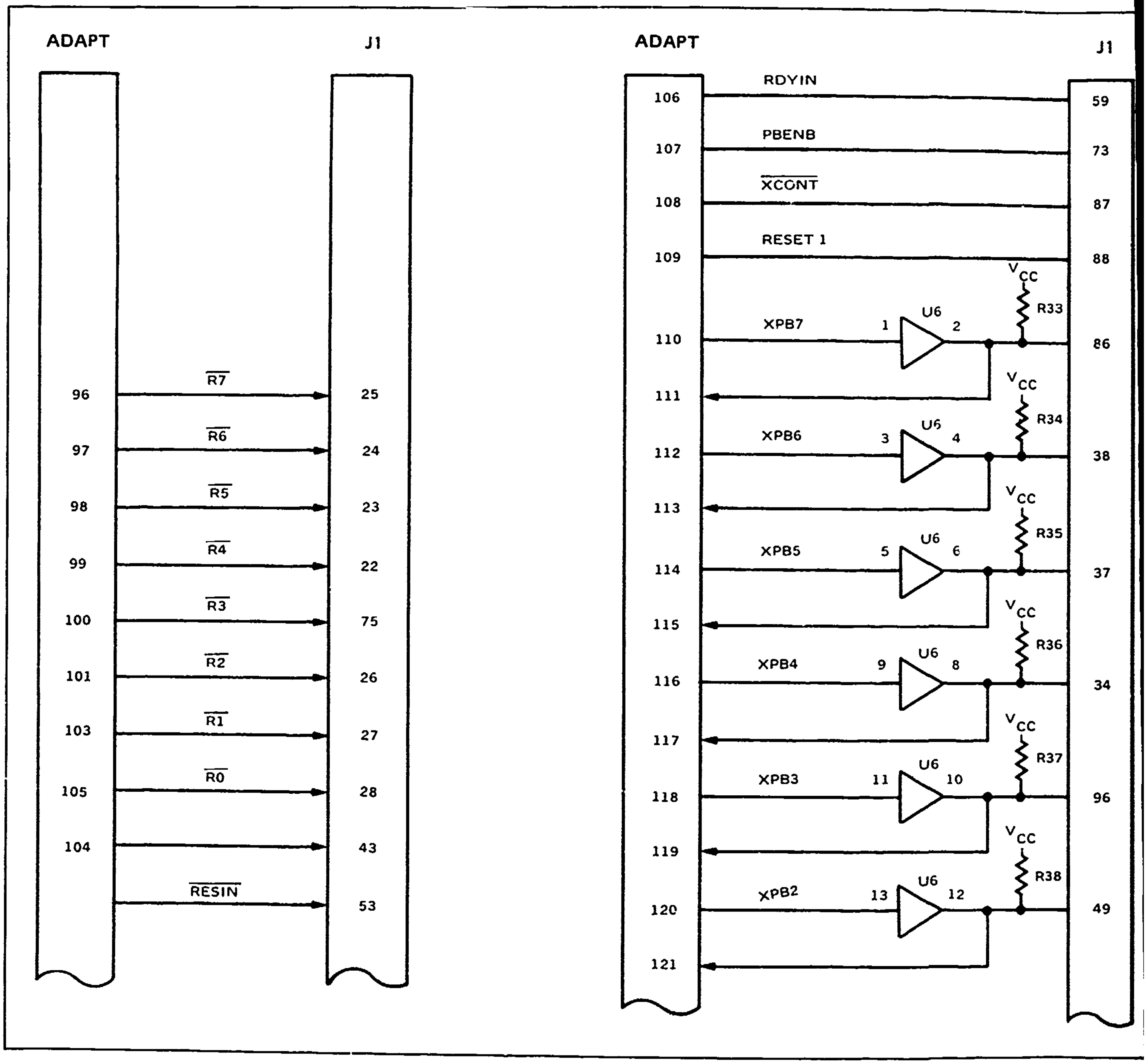
SACMPR - INIT INITIALIZATION

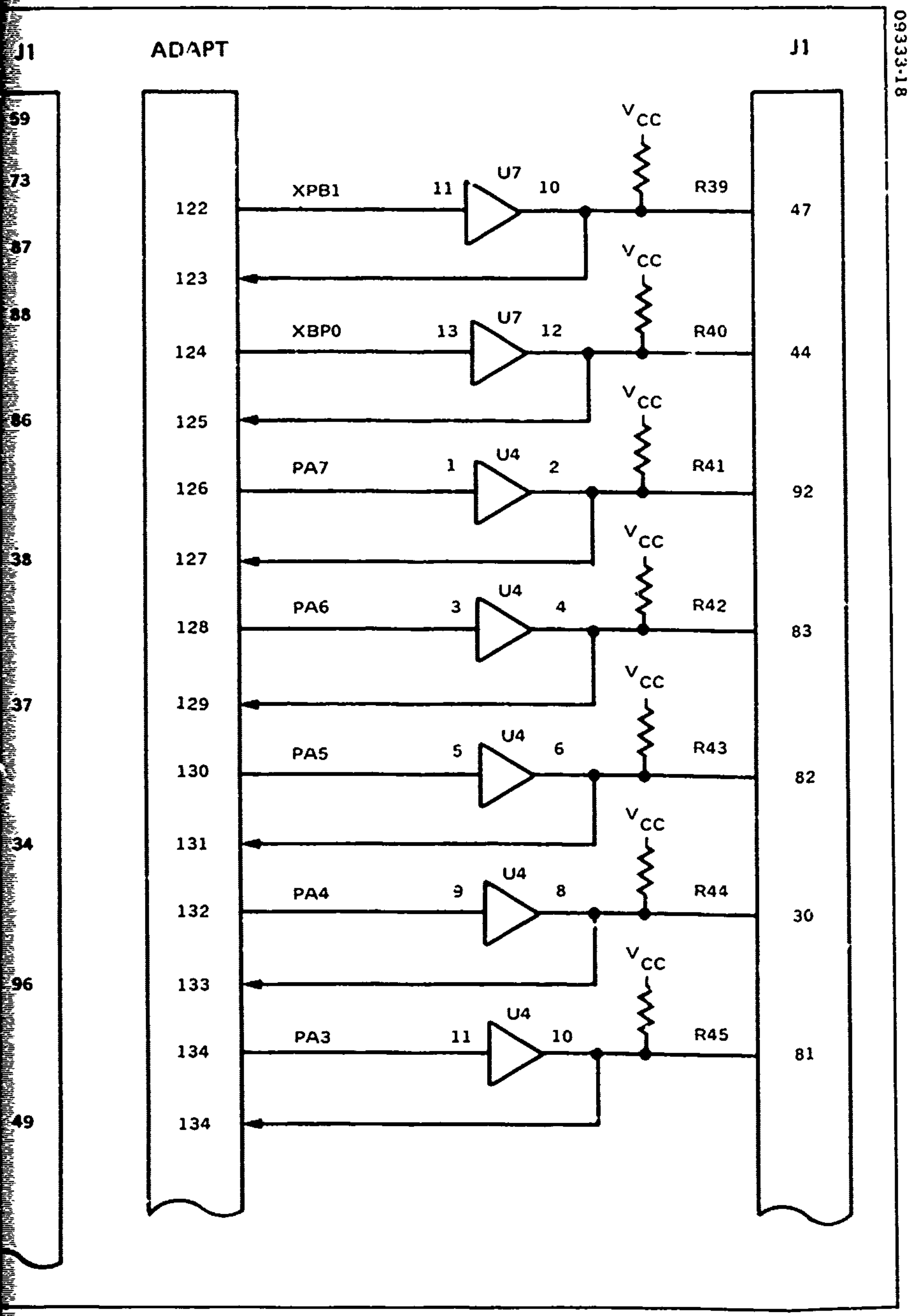
09333-17

2

APPENDIX B – Schematics
Section B. 2 – PN 1635972 Circuit Board Test Adapter

B. 2. 4 – 8255 PPI I/O BUFFER AND PULL UP; PART 4 OF 6

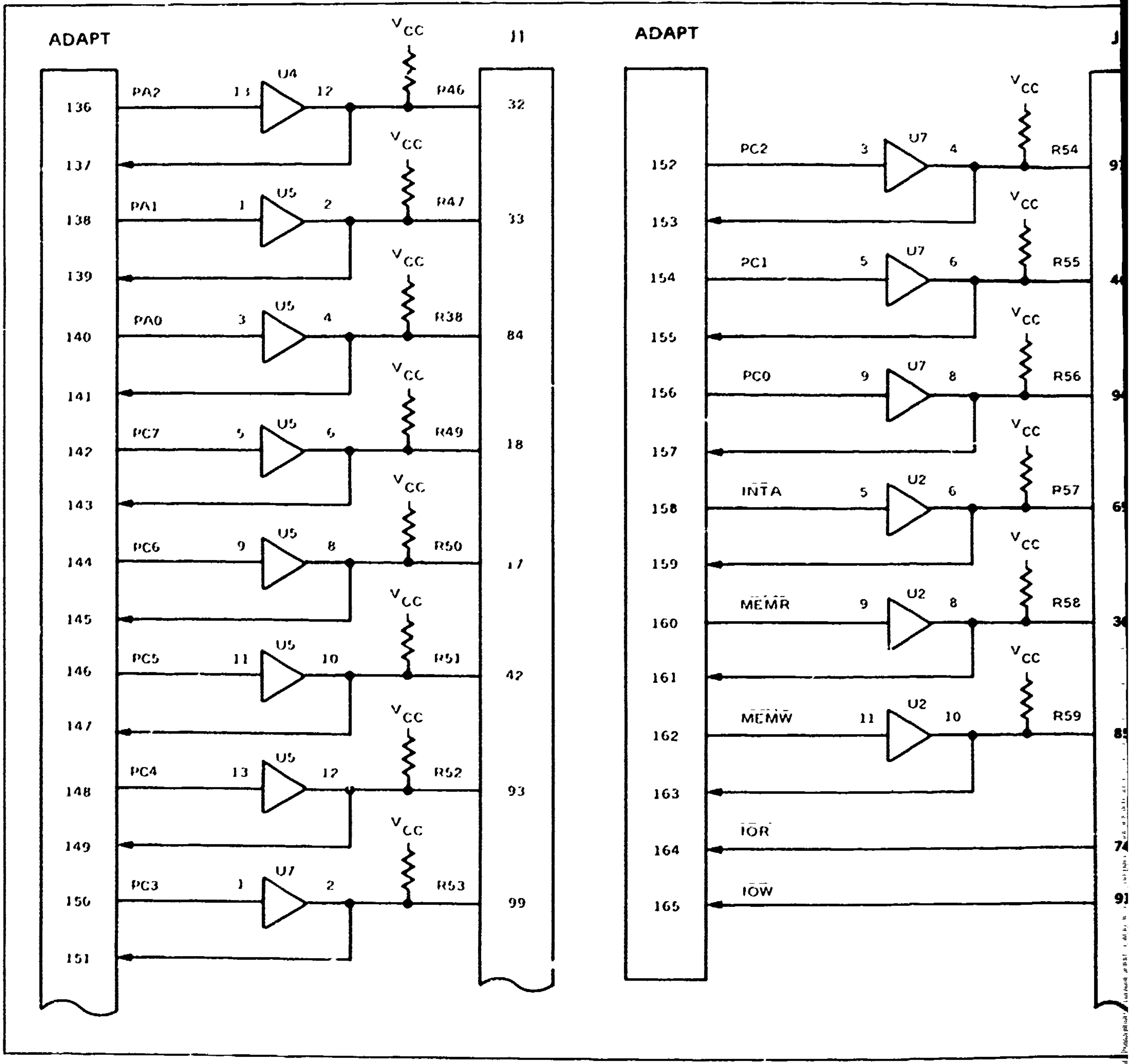


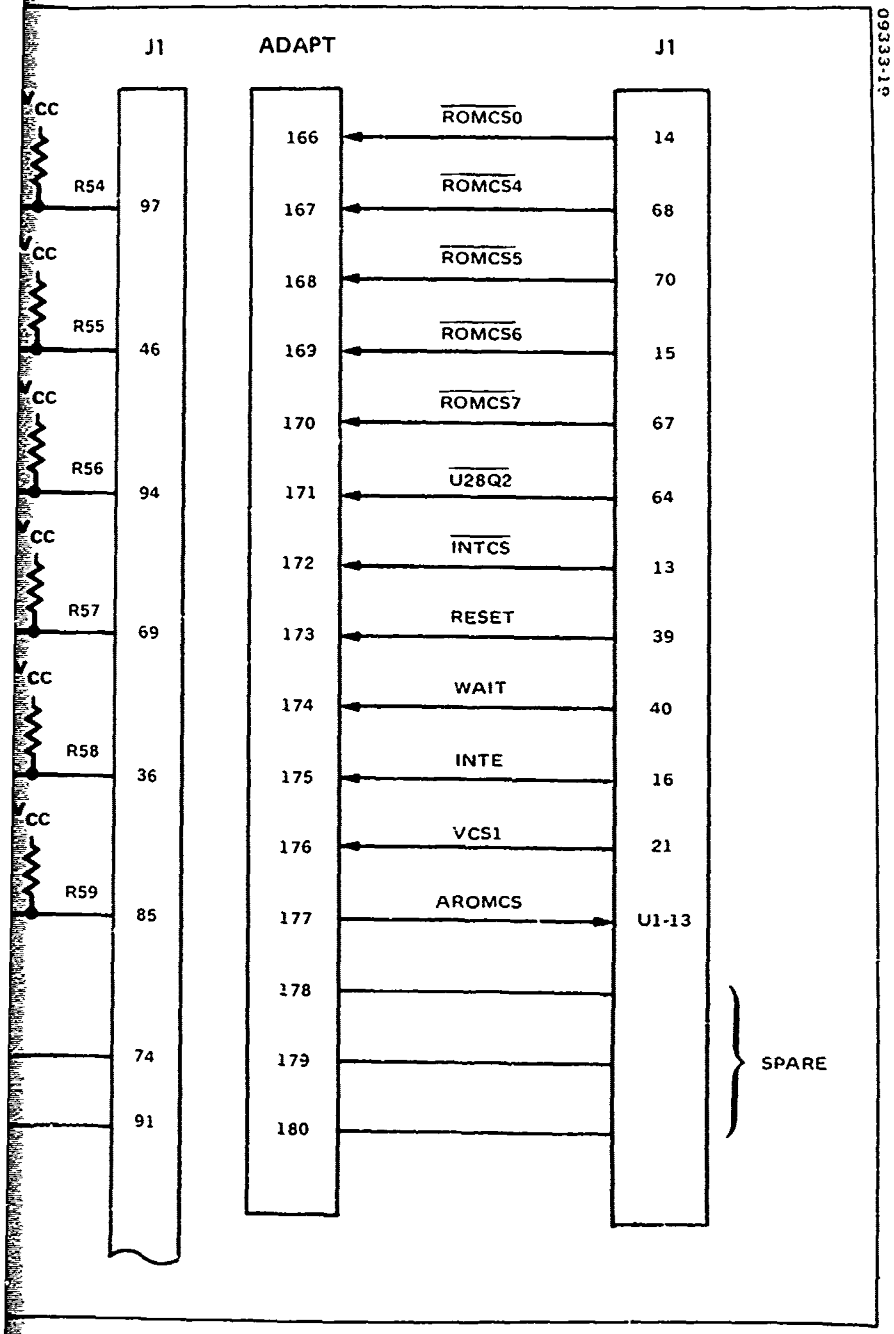

09333-18
J1
ADAPT
J1
59
73
87
88
86
38
37
34
96
49
122
123
124
125
126
127
128
129
130
131
132
133
134
134
XPB1
XBP0
PA7
PA6
PA5
PA4
PA3
U7
U7
U4
U4
U4
U4
U4
11
10
13
12
1
2
3
4
5
6
9
8
11
10
VCC
R39
R40
R41
R42
R43
R44
R45
47
44
92
83
82
30
81

2

APPENDIX B – Schematics
Section B. 2 – PN 1635972 Circuit Board Test Adapter

B. 2. 5 – 8255 PPI I/O BUFFER AND PULL UP; PART 5 OF 6

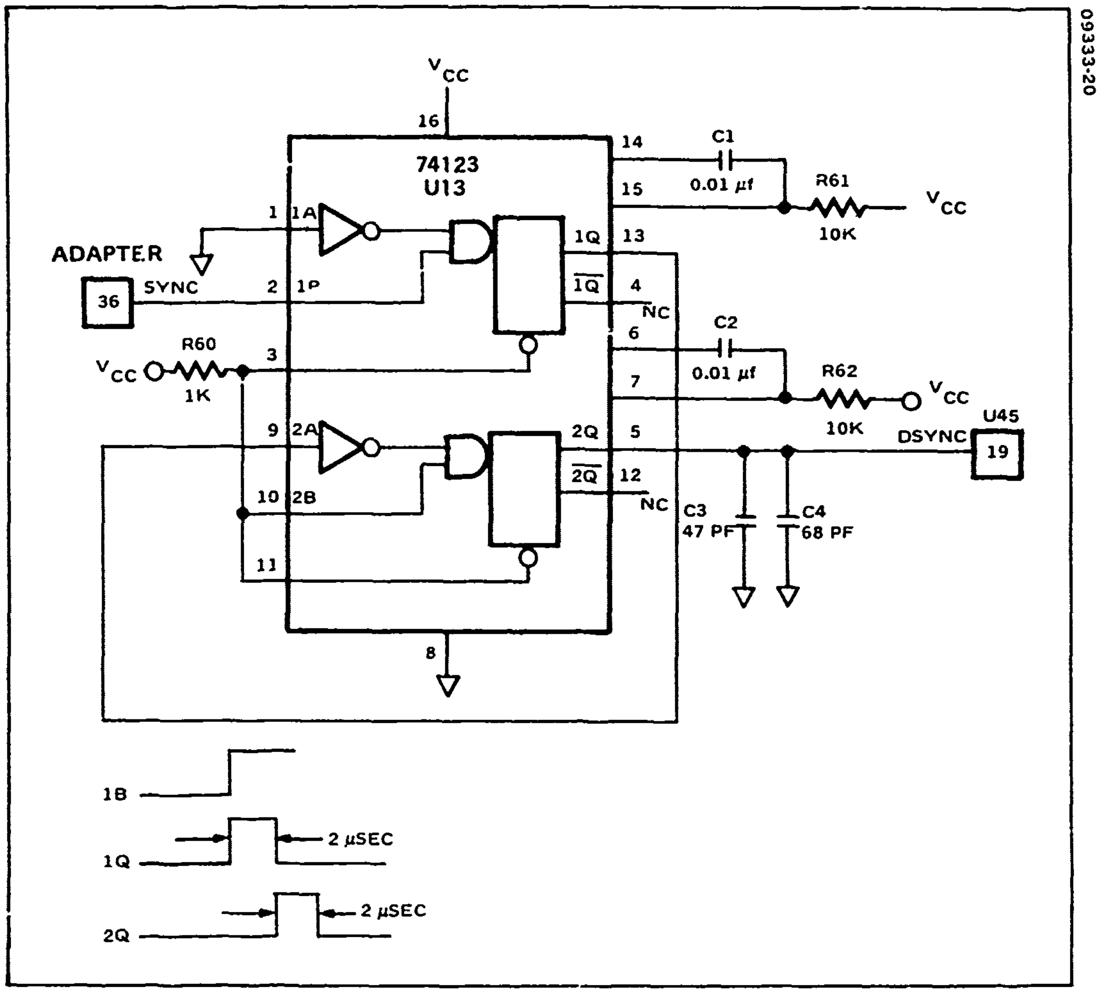



09333-19
J1
ADAPT
J1
VCC
R54
97
VCC
R55
46
VCC
R56
94
VCC
R57
69
VCC
R58
36
VCC
R59
85
74
91
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
ROMCS0
ROMCS4
ROMCS5
ROMCS6
ROMCS7
U28Q2
INTCS
RESET
WAIT
INTE
VCS1
AROMCS
14
68
70
15
67
64
13
39
40
16
21
U1-13
SPARE

2

1

B.2.6 – 8228 SCBD, STROBE DELAY CIRCUIT; PART 6 OF 6



B.2.7 – TEST ADAPTER PARTS LIST

Parts List

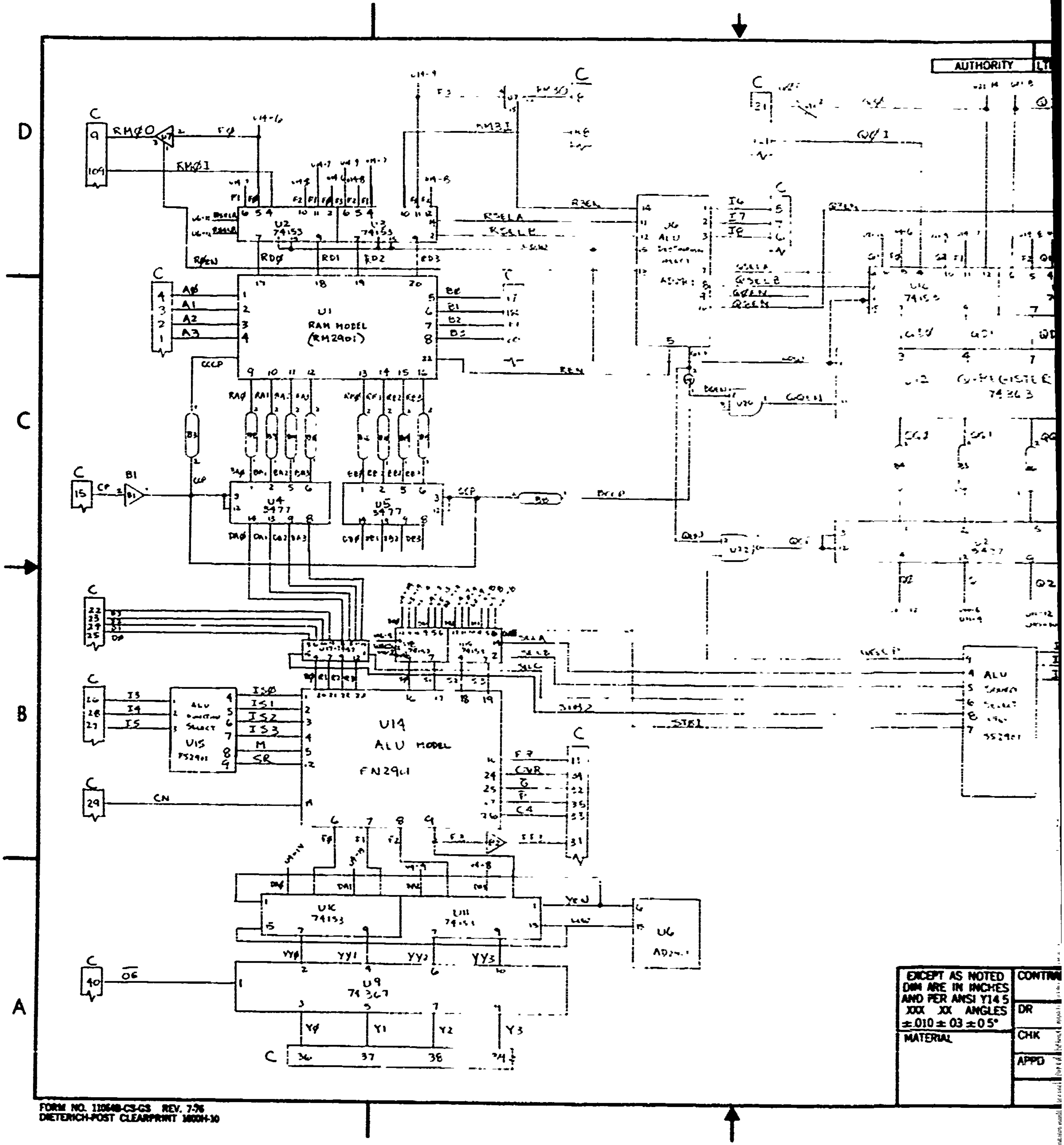
Ref. Designator	Part Number	Quantity
U1 - U10	7417	10
U13	74123	1
U11 - U12	93446 (ROM)	2
C3	47 Pf	1
C4	68 Pf	1
R1 - R60	1K (1/4W, 5%)	60
C1, C2	0.01µf	2
R61, R62	10K (1/4W, 5%)	2
DS1 - DS17	LED - 547-2007	17
DIP SOCKET	16 pin - AUGAT-D	20

SECTION B. 3

PN 1646178 CIRCUIT BOARD MODEL

APPENDIX B – Schematics
Section B. 3 – PN 1646178 Circuit Board Model

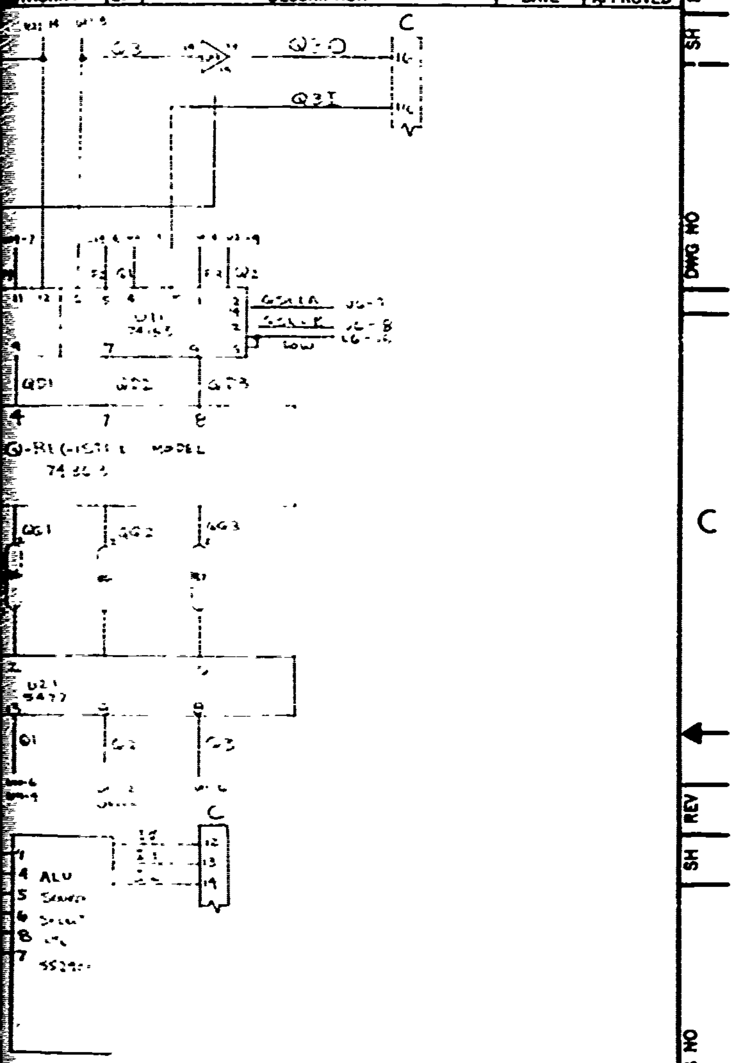
B. 3. 1 – AM2901 MODEL



FORM NO. 11064B-CS-GS REV. 7-76
DIETERICH-POST CLEARPRINT 1000H-10

09333-4

	REVISIONS			
THORITY	LTR	DESCRIPTION	DATE	APPROVED



C

REV

SH

DWG NO

REV

SH

DWG NO

S NOTED N INCHES NSI Y14.5 ANGLES ±0.5°	CONTRACT	HUGHES	HUGHES AIRCRAFT COMPANY FULLERTON, CALIFORNIA
	DR	[illegible]	
	CHK		
	APPD	SIZE C	FSCM NO 05869 · DWG NO · REV
		SCALE	SHEET

A

2

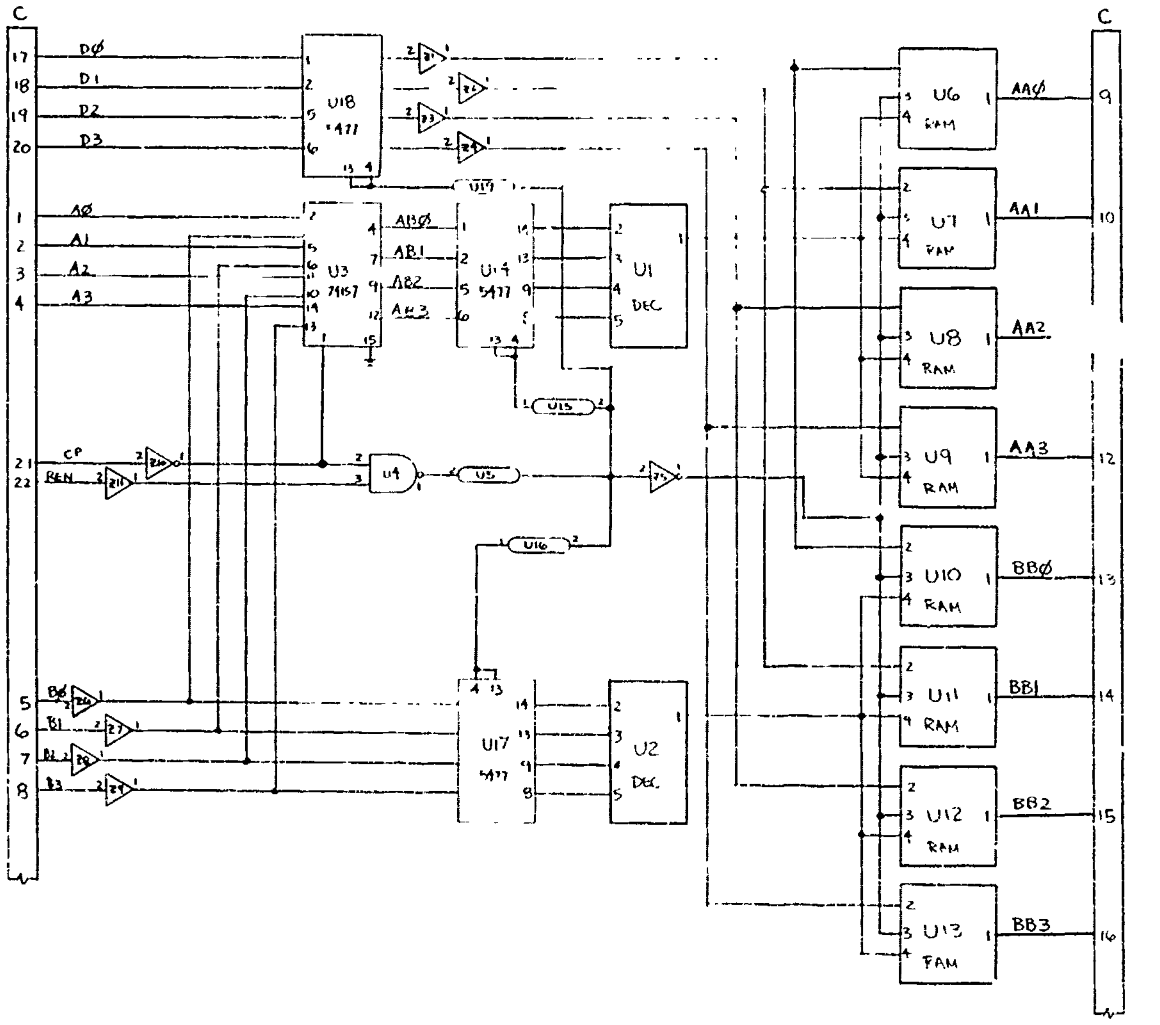
1

APPENDIX B – Schematics
Section B. 3 – PN 1646178 Circuit Board Model

B. 3. 2 – AM2901, RAM MODEL

AUTHORITY

D



C

B

A

EXCEPT AS NOTED
DIM ARE IN INCHES
AND PER ANSI Y14.5
XXX XX ANGLES
± 010 ± 03 ± 0 5°

MATERIAL

FORM NO. 11064B-CS-G3 REV 7-76
DIETERICH-POST CLEARPRINT 1000H-10

REVISIONS				
THORITY	LTR	DESCRIPTION	DATE	APPROVED

09333-5

REV

SH

DWG NO

C

REV

SH

DWG NO

NOTED INCHES I Y14 5 NGLES ±0 5°	CONTRACT	HUGHES	HUGHES AIRCRAFT COMPANY FULLERTON, CALIFORNIA		
	DR	AM 2901 RAM MODEL			
	CHK				
	APPD	SIZE C	FSCM NO 05869	DWG NO	REV
		SCALE		SHEET	

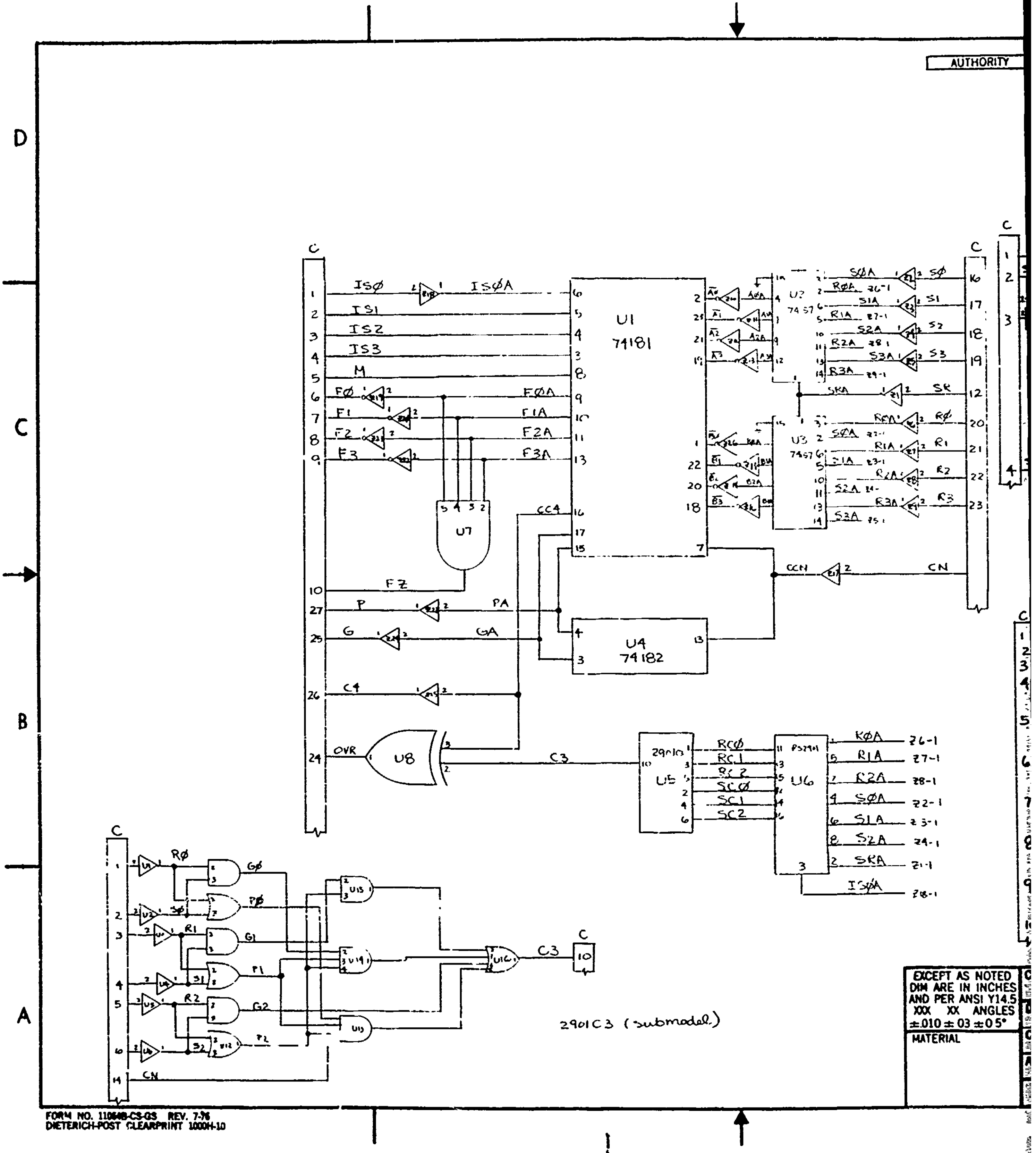
A

1

2

APPENDIX B – Schematics
Section B. 3 – PN 1646178 Circuit Board Model

B. 3. 2 – AM 2901, ALU MODEL

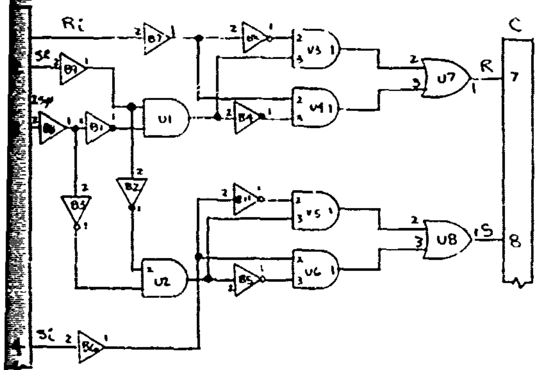


REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED

REV

SH

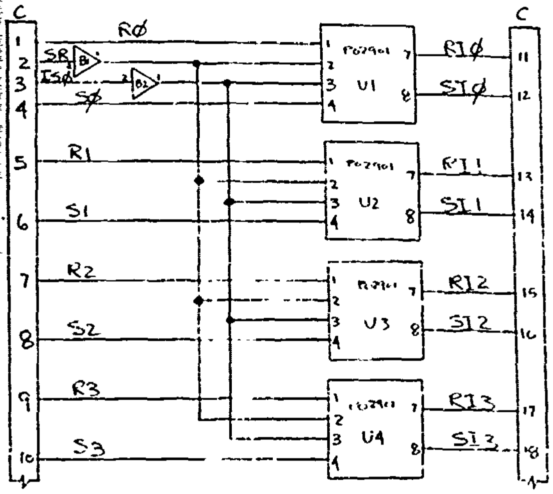
PO2901 (SUBMODEL)



DWG NO

C

RS2901 (submodel)



REV

SH

DWG NO

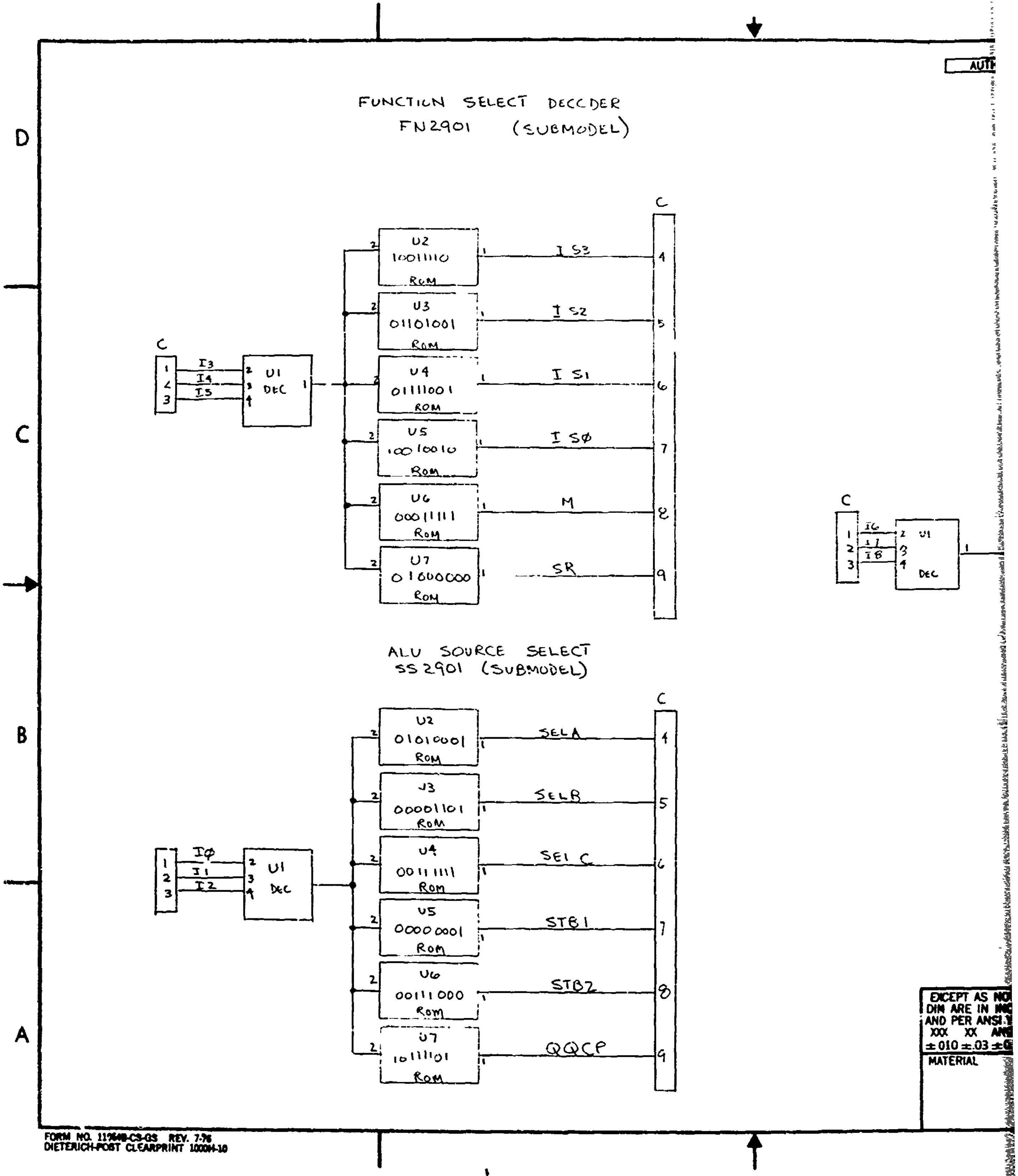
CONTRACT	HUGHES	HUGHES AIRCRAFT COMPANY FULLERTON, CALIFORNIA			
DR	ALU MODEL, AM2901				
CHK					
AFPD	SIZE C	FSCM NO 05869	DWG NO		REV
	SCALE		SHEET		

A

2

APPENDIX B – Schematics
Section B. 3 – PN1646178 Circuit Board Model

B. 3. 4 – AM 2901, MICROINSTRUCTION DECODER



FORM NO. 1196-CS-GS REV. 7-76
DIETERICH-POST CLEARPRINT 1000H-10

09333-7

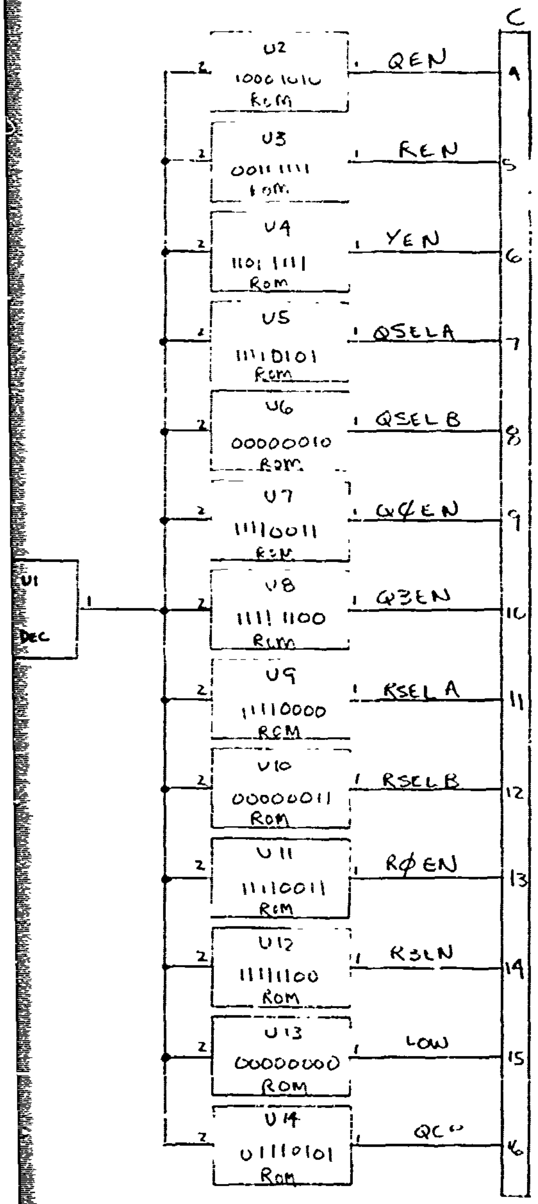
REVISIONS				
AUTHORITY	LTR	DESCRIPTION	DATE	APPROVED

REV

SH

DWG NO

ALU DESTINFTION SELCCT
AD 2901 (SUBMCDEL)



C

REV

SH

DWG NO

EXCEPT AS NOTED DIM ARE IN INCHES AND PER ANSI Y14 5 .XXX .XX ANGLES ±.010 ±.03 ±0 5° MATERIAL	CONTRACT	HUGHES	HUGHES AIRCRAFT COMPANY FULLERTON, CALIFORNIA		
	DR	AM2901, MICROINSTR DECODE			
	CHK				
	APPD	SIZE C	FSCM NO 05869	DWG NO	REV
		SCALE		SHEET	

A

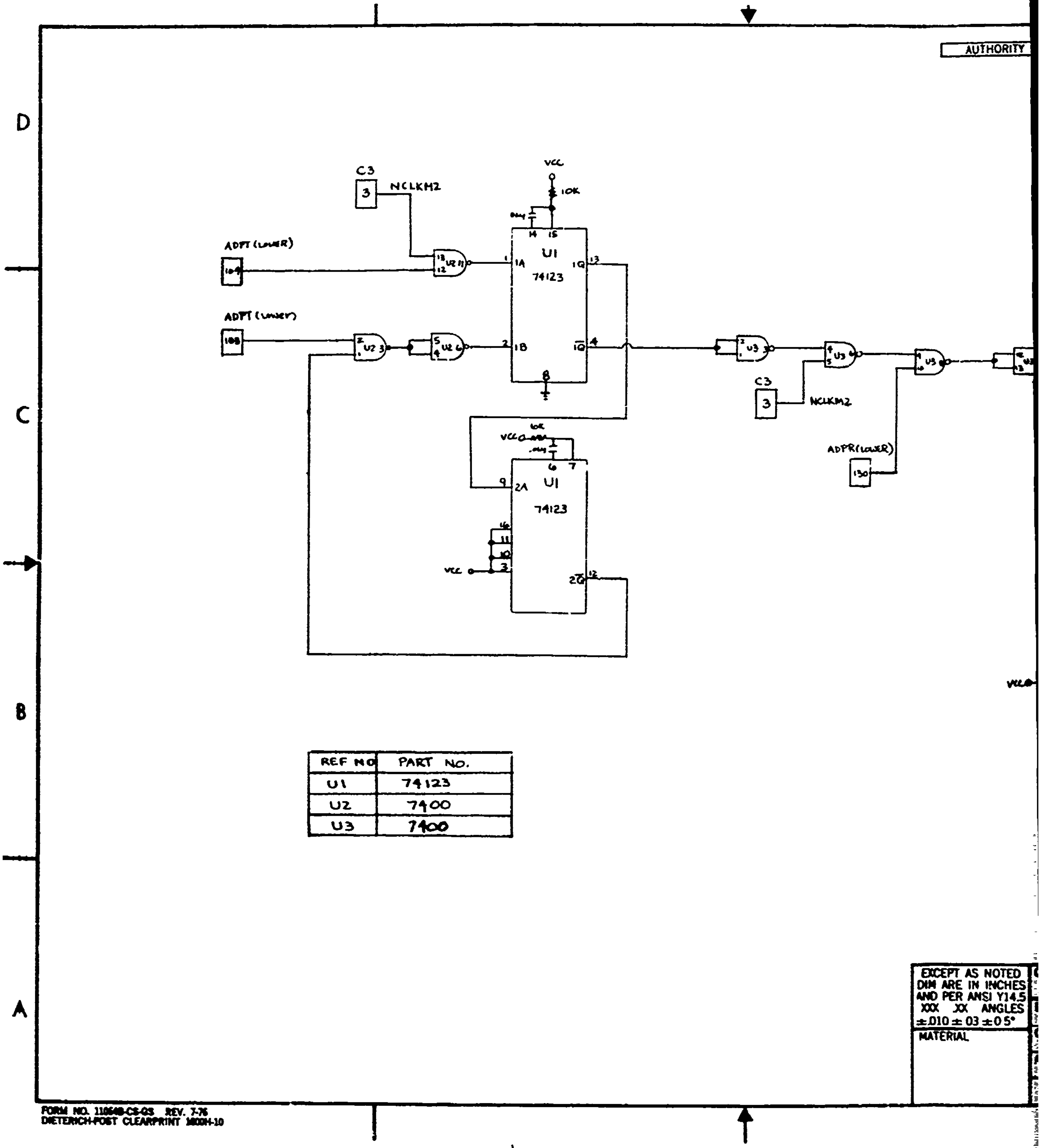
2

SECTION B. 4

PN 1646178 CIRCUIT BOARD TEST ADAPTER

APPENDIX B – Schen:atics
Section B. 4 – PN 1646178 Circuit Board Test Adapter

B. 4. 1 – INITIALIZATION CIRCUIT



REF NO	PART NO.
U1	74123
U2	7400
U3	7400

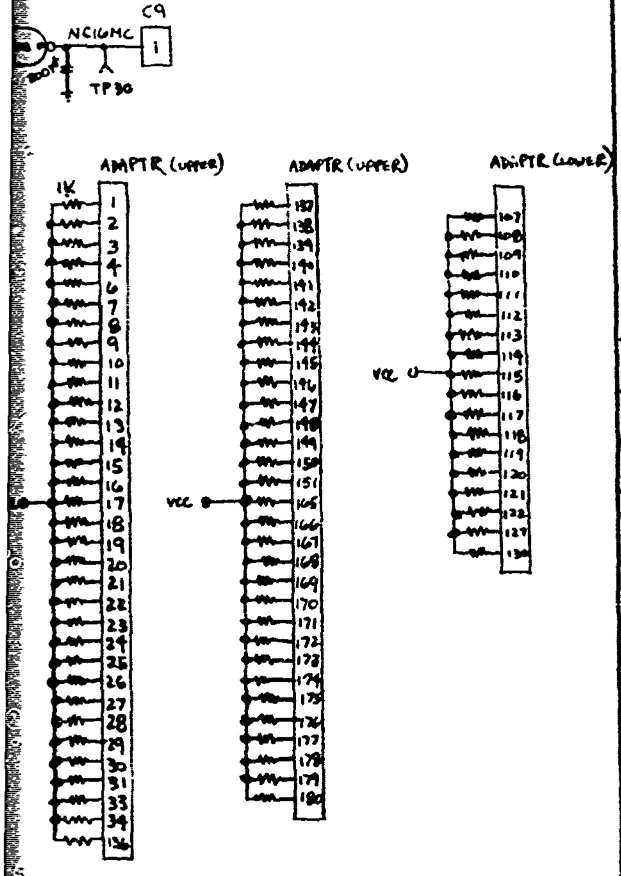
EXCEPT AS NOTED
DIM ARE IN INCHES
AND PER ANSI Y14.5
XXX XX ANGLES
±.010 ± 03 ± 0 5°

MATERIAL

FORM NO. 11064B-CS-GS REV. 7-76
DIETERICH-POST CLEARPRINT 1000H-10

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED

09333-8



C

A

CONTRACT	HUGHES	HUGHES AIRCRAFT COMPANY FULLERTON, CALIFORNIA
DR	INITIALIZATION CKT	
CHK		
APPD	SIZE C	FSCM NO 05869	DWG NO	REV
	SCALE		SHEET

2

1

APPENDIX B – Schematics
Section B.4 – PN 1646178 Circuit Board Test Adapter
B.4.2 – WIRE LIST, ADAPTER, PART 1

AUTHORITY | L

J10		C10	ADPT	J9		C9	ADPT	J8		C8
1	NOPS13	1	★180	1	NCLKMC		[illegible]			1
2	NOPS14	2	★ 79	2		2	[illegible]	2		2
3	NOPS15	3	★178	3	CLKMO	3	[illegible]	3	NX2054	3
4	NXRLAD	4	★177	4	NX2054	4	[illegible]	4		4
5	NOMCKC	5	★176	5	NXBER4	5	[illegible]	5	XSITSW	5
6	QRAR04	6	★175	6	NXBER5	6	[illegible]	6		6
7	QRAR05	7	★174	7	NXBER1	7	★124	7	XJMP02	7
8	QRAR06	8	★173	8	NXBER2	8	★123	8		8
9	QRAR07	9	★172	9	XDB00	9	[illegible]	9	XJMP01	9
10	QRAR08	10	★ 71	10	XDB01	10	★121	10		10
11	QRAR09	11	★170	11	XDB02	11	★120	11	XRDACK	11
12	QRAR10	12	★169	12	XDB03	12	★119	12		12
13	QRAR11	13	★168	13	XDB04	13	★118	13	XWTACK	13
14	QRAR12	14	★167	14	XDB05	14	★117	14		14
15	QRAR13	15	★166	15	XDB06	15	★116	15	XPLTSW	15
16	QRAR14	16	★165	16	XDB07	16	★115	16		16
17	QRAR15	17	★164	17	XDB08	17	★114	17	XEM	17
18	XDRD00	18	★163	18	XDB09	18	★113	18		18
19	XDRD01	19	★162	19	XDB10	19	★112	19	XEM2	19
20	XDRD02	20	★161	20	XDB11	20	★111	20		20
21	XDRD03	21	★160	21	XDB12	21	★110	21	XEM3	21
22	XDRD04	22	★159	22	XDB13	22	★109	22		22
23	XDRD05	23	★158	23	XDB14	23	★108	23	XREAD	23
24	XDRD06	24	★157	24	XDB15	24	★107	24		24
25	XDRD07	25	★156	25	QOF1	25	★106	25	XWRITE	25
26	XDRD08	26	★155	26	QOF2	26	★105	26		26
27	XDRD09	27	★154	27	QOF3	27	94	27	XADDR	27
28	XDRD10	28	★153	28	QOF4	28	95	28		28
29	XDRD11	29	★152	29	QOF5	29	96	29	XJMP03	29
30	XDRD12	30	★151	30	QOF6	30	[illegible]	30	XIF13	30
31	XDRD13	31	★150	31	NXMC	31	[illegible]	31	XEM4	31
32	XDRD14	32	★149	32	XIF1	32	[illegible]	32	XEM5	32
33	XDRD15	33	★ 45	33	XIF2	33	[illegible]	33	NXRDLD	33
34	XDRD16	34	★ 47	34	XIF3	34	[illegible]	34		34
35	XDRD17	35	★ 46	35	XIF4	35	[illegible]	35	XMB15	35
36	XDRD18	36	★145	36	XIF5	36	[illegible]	36	XMB14	36
37	XDRD19	37	★144	37	XIF6	37	[illegible]	37	XMB13	37
38	XDRD20	38	★143	38	XIF7	38	[illegible]	38	XMB12	38
39	XDRD21	39	★142	39	XIF8	39	120	39	XMB11	39
40	XDRD22	40	★141	40	XIF9	40	[illegible]	40	XMB10	40
41	XDRD23	41	★140	41	XIF10	41	[illegible]	41	XMB09	41
42	XDRD24	42	★139	42	XIF11	42	[illegible]	42	XMB08	42
43	XDRD25	43	★ 38	43	XJMP01	43	[illegible]	43	XMB07	43
44	XDRD26	44	★137	44	XJMP02	44		44	XMB06	44
45	XDRD27	45	★136	45	XJMP03	45	12	45	XMB05	45
46	XDRD28	46	★135	46	NXBER3	46	13	46	XMB04	46
47	XDRD29	47	★134	47	NXER2	47	14	47	XMB03	47
48	XDRD30	48	★133	48	NXER3	48	15	48	XMB02	48
49	XDRD31	49	★132	49	NXER4	49	[illegible]	49	XMB01	49
50	QERR	50	★131	50	NXER5	50	[illegible]	50	XMB00	50

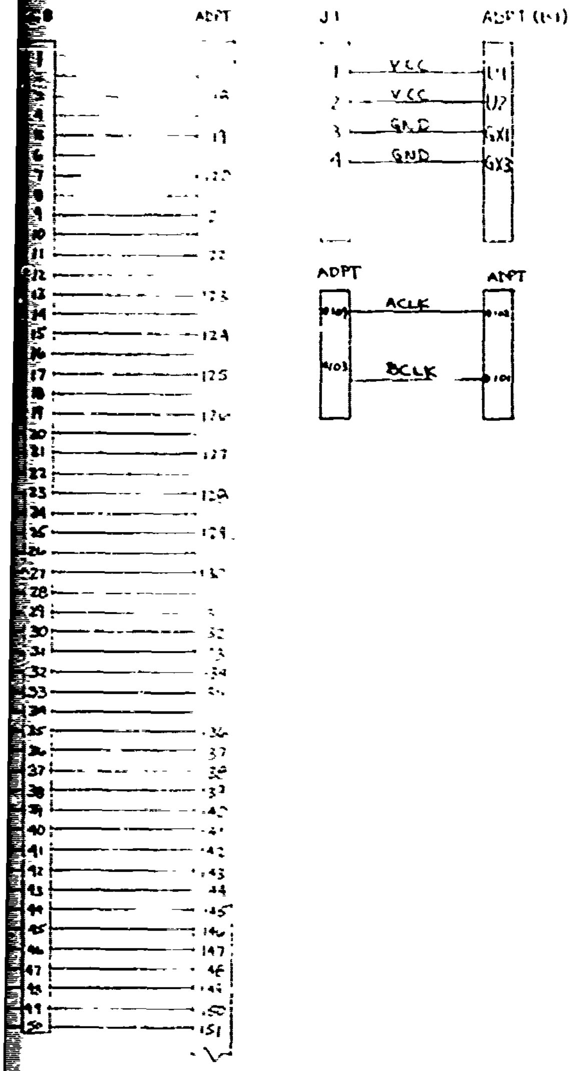
NOTE: ★ Designates Pin on Lower Adapter Board

EXCEPT AS NOTED DIM ARE IN INCHES AND PER ANSI Y14.5 XXX XX ANGLES ±.010 ±.03 ±0.5°	CONT
	DR
MATERIAL	CHK
	APPD

FORM NO. 110640-CS-03 REV. 7-76
DIETERICH-POST CLEARPRINT 1000H-10

		REVISIONS			
HORITY	LTR	DESCRIPTION	DATE	APPROVED	REV

09333-9



SH

REV

DWG NO

C

SH

REV

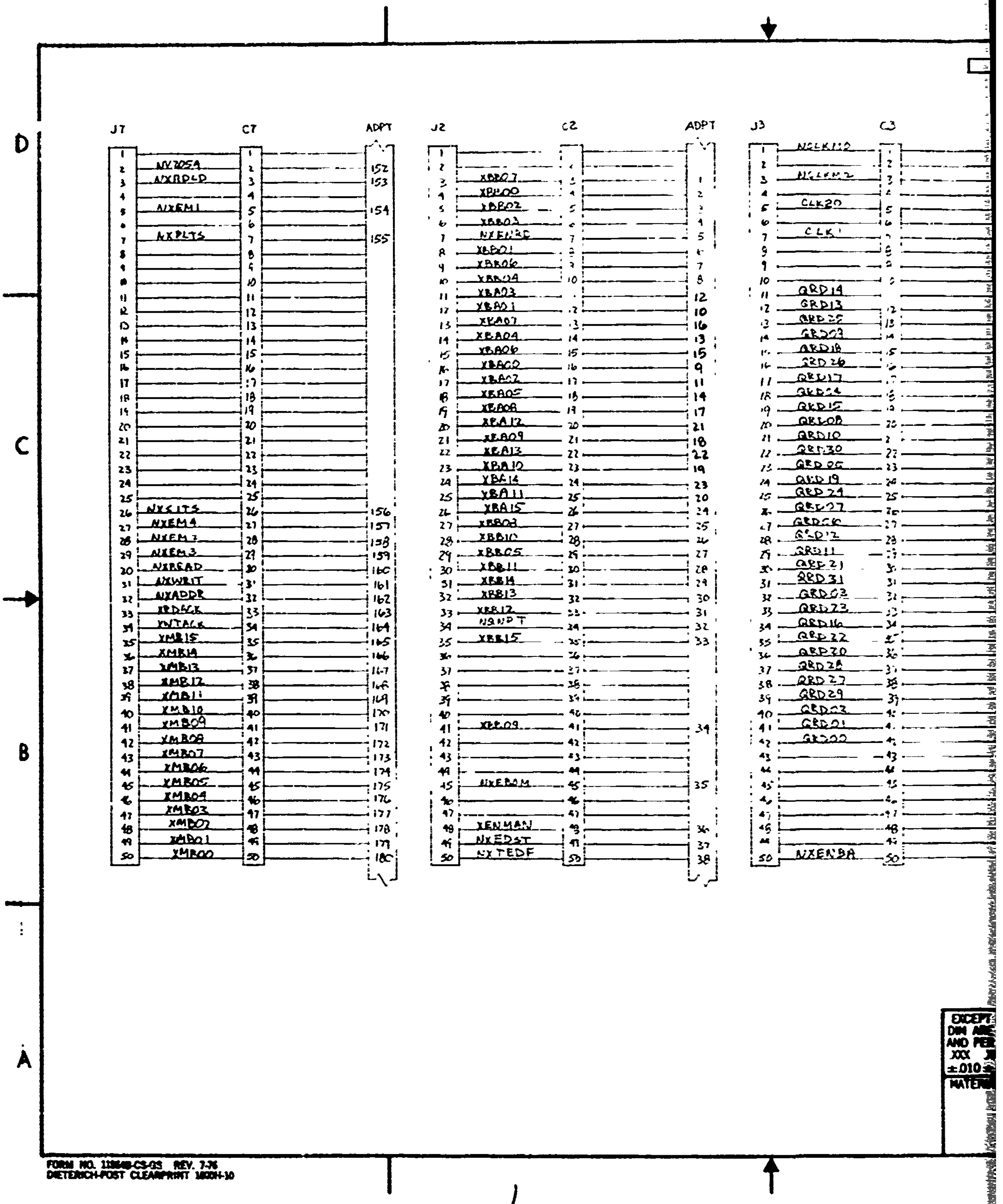
DWG NO

NOTED IN INCHES ANSI Y14 5 ANGLES ±0 5°	CONTRACT	HUGHES	HUGHES AIRCRAFT COMPANY FULLERTON CALIFORNIA	
	DR E Levitch	WIRE LIST, ADAPTER		
	CHK			
	APPD	SIZE C	FSCM NO 05869	DWG NO	REV
	4-23-80	SCALE		SHEET 1 OF 2

A

APPENDIX B – Schematics
Section B. 4 – PN 1646178 Circuit Board Test Adapter

B. 4. 3 – WIRE LIST, ADAPTER, PART 2



J7		C7	ADPT
1		1	
2	NV205A	2	152
3	NXRDLD	3	153
4		4	
5	NXEM1	5	154
6		6	
7	NXPLTS	7	155
8		8	
9		9	
10		10	
11		11	
12		12	
13		13	
14		14	
15		15	
16		16	
17		17	
18		18	
19		19	
20		20	
21		21	
22		22	
23		23	
24		24	
25		25	
26	NXSITS	26	156
27	NXEM4	27	157
28	NXEM2	28	158
29	NXEM3	29	159
20	NXREAD	30	160
31	NXWRIT	31	161
32	NXADDR	32	162
33	XRDACK	33	163
34	XWTACK	34	164
35	XMB15	35	165
36	XMB14	36	166
37	XMB13	37	167
38	XMB12	38	168
39	XMB11	39	169
40	XMB10	40	170
41	XMB09	41	171
42	XMB08	42	172
43	XMB07	43	173
44	XMB06	44	174
45	XMB05	45	175
46	XMB04	46	176
47	XMB03	47	177
48	XMB02	48	178
49	XMB01	49	179
50	XMB00	50	180

J2		C2	ADPT
1			
2		2	
3	XBB07	3	1
4	XBB00	4	2
5	XBB02	5	3
6	XBB03	6	4
7	NXENBC	7	5
8	XBB01	8	6
9	XBB06	9	7
10	XBB04	10	8
11	XBA03		12
12	XBA01	12	10
13	XBA07	13	16
14	XBA04	14	13
15	XBA06	15	15
16	XBA00	16	9
17	XBA02	17	11
18	XBA05	18	14
19	XBA08	19	17
20	XBA12	20	21
21	XBA09	21	18
22	XBA13	22	22
23	XBA10	23	19
24	XBA14	24	23
25	XBA11	25	20
26	XBA15	26	24
27	XBB08	27	25
28	XBB10	28	26
29	XBB05	29	27
30	XBB11	30	28
51	XBB14	31	29
32	XBB13	32	30
33	XBB12	33	31
34	NSNPT	34	32
35	XBB15	35	33
36		36	
37		37	
38		38	
39		39	
40		40	
41	XBB09	41	34
42		42	
43		43	
44		44	
45	NXEBAM	45	35
46		46	
47		47	
48	XENMAN	48	36
49	NXEDST	49	37
50	NXTEDF	50	38

J3		C3
1	NCLKI12	1
2		2
3	NCLKM2	3
4		4
5	CLK20	5
6		6
7	CLK1	7
8		8
9		9
10		10
11	GRD14	
12	GRD13	12
13	GRD25	13
14	GRD09	14
15	GRD18	15
16	GRD26	16
17	GRD17	17
18	GRD04	18
19	GRD15	19
20	GRD08	20
21	GRD10	21
22	GRD30	22
23	GRD05	23
24	GRD19	24
25	GRD24	25
26	GRD07	26
27	GRD06	27
28	GRD12	28
29	GRD11	29
30	GRD21	30
31	GRD31	31
32	GRD03	32
33	GRD23	33
34	GRD16	34
35	GRD22	35
36	GRD20	36
37	GRD28	37
38	GRD27	38
39	GRD29	39
40	GRD02	40
41	GRD01	41
42	GRD00	42
43		43
44		44
45		45
46		46
47		47
48		48
49		49
50	NXENBA	50

FORM NO. 118648-CS-GS REV. 7-76
DIETERICH-POST CLEARPRINT 1000H-10

REVISIONS				
AUTHORITY	LTR	DESCRIPTION	DATE	APPROVED

09333-10

ADPT	J4		C4	ADPT
39	1		1	
	2		2	
40	3		3	
	4		4	
41	5		5	
	6		6	
42	7		7	
	8		8	
	9		9	
	10	QG	10	76
57	11		11	
56	12		12	
68	13		13	
52	14		14	
61	15		15	
69	16		16	
60	17		17	
47	18		18	
58	19		19	
51	20		20	
53	21	NXTCKAI	21	77
73	22	QCSV	22	76
48	23	QDMCCA	23	79
62	24		24	
67	25	NGMCRH	25	80
50	26	NXBCM2	26	61
49	27		27	
55	28		28	
54	29		29	
64	30		30	
74	31		31	
46	32	QCVF	32	82
66	33		33	
59	34	NXOSCI	34	83
65	35	QL	35	84
63	36		36	
71	37	NXRCKA	37	85
70	38	KTINT	38	86
72	39	NXLRXZ	39	87
45	40		40	
44	41		41	
43	42	NCKEXT	42	88
	43	QFSYN	43	89
	44		44	
	45	NGMCC	45	90
	46		46	
	47	NCKERS	47	91
	48		48	
	49	NXOMCM	49	92
15	50	NXERSP	50	93

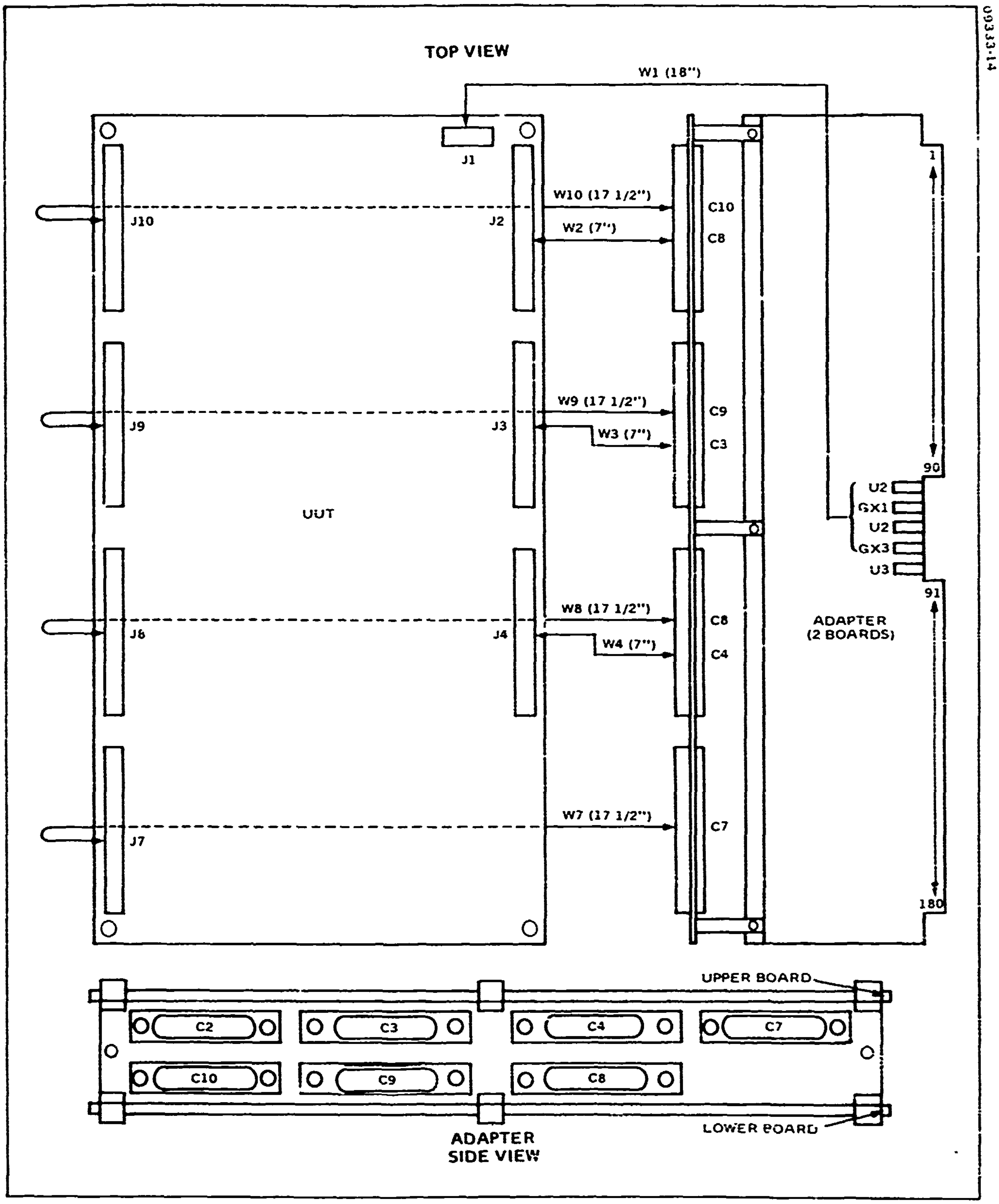
C

A

EXCEPT AS NOTED DIM ARE IN INCHES AND PER ANSI Y14.5 .XXX .XX ANGLES ±.010 ±.03 ±0.5°	CONTRACT	HUGHES	HUGHES AIRCRAFT COMPANY FULLERTON CALIFORNIA		
	DR E. Leitch	WIRE LIST, ADAPTER			
MATERIAL	CHK				
	APPD	SIZE C	FSCM NO 05869	DWG NO	REV
	9-23-80	SCALE		SHEET 2 OF 2	

2

B.4.4 – LAYOUT AND INTER CONNECTIONS. ADAPTER



APPENDIX C – REVISIONS

SECTION C.1

REVISIONS TO DTS-70 IMPLEMENTATION PLAN

APPENDIX C – Revisions
Section 1 – Revisions to DTS 70 Implementation Plan

C.1.1 DTS 70 IMPLEMENTATION PLAN CDRL A006, 21 JULY 1980, REVISIONS

REVISIONS

Graph II, page 13:

The data point must be revised for the MC 8080 A/B microprocessor using Signature Analysis with the DTS-70 system at Hughes Fullerton.

Later information revealed that test program time required through hardware verification increased. In addition a correction in the number of ICs for the 1635972 D/PCB is required which includes the MC 8080 A/B microprocessor. Therefore the data point changes as follows:

Item	Was	Is
MC 8080 A/B	38 IC	28 IC
Hughes Fullerton	14 Man Weeks	23 Man Weeks

Page 14 Paragraph B:

General Dynamics DTS-70; 8085

Was: "In the GD DTS-70 data point case, the 8085 was modeled directly with logic primitives."

Is: "In the GD DTS-70 data point case, the 8085 was functionally modeled."

This information was received during the Industry Demonstration from a representative of General Dynamics, Pomona, Ca.

REVISIONS

Page 14 Paragraph C:

HAC (DFI) DTS-70; 8080

Was: "Using the test technique outlined total programming time for this PCB on the DTS-70 required 14 man weeks. For the same PCB (38 ICs including the 8080), the GR-195 programming time is estimated at 19 man weeks or 36 percent longer."

Is: "Using the test technique outlined total programming time for this PCB on the DTS-70 required 23 man weeks. Relative to the GD 8085 data point, it is observed that the Signature Analysis functional test of the 8080 vs. functional modeling of the 8085 achieves a reduction in test program time in the ratio of 23/45 or very nearly 1:2.

Page 15 Paragraph E:

DTS-70; 2901

Subparagraph 3

Was: '. substantial reduction----functionally tested instead."

Is: "An 8080 type LSI device when functionally tested using Signature Analysis can substantially reduce test programming time by a ratio approaching 1:2 as compared to a functionally modeled 8085 test program."